A
Dissertation
on

Design and Implementation of Semantic Annotation Tool with Ontology Creation and Semi-automatic Ontology Population

Submitted in partial fulfillment of the requirement
For the award of Degree of

MASTER OF ENGINEERING
(Computer Technology and Application)

Submitted by

ANUJ KUMAR
(University Roll No. 10183)

Under the Guidance of
Dr. DAYA GUPTA

DEPARTMENT OF COMPUTER ENGINEERING
DELHI COLLEGE OF ENGINEERING
BAWANA ROAD, DELHI
(DELHI UNIVERSITY)
June-2008

CERTIFICATE

DELHI COLLEGE OF ENGINEERING
(Govt. of National Capital Territory of Delhi)
BAWANA ROAD, DELHI - 110042

Date:_____

This is to certify that project entitled "**Design and Implementation of Semantic Annotation Tool with Ontology Creation and Semi-automatic Ontology Population**" has been completed by Mr. **Anuj Kumar** in partial fulfilment of the requirement of **Master in Engineering** in **Computer Technology & Application**.

This is a record of his work carried out by him under my supervision and support. This is a beneficial work in field of Semantic Web for creating a semantic web infrastructure using the currently available technologies and resources.

(Dr. DAYA GUPTA)
HOD & PROJECT GUIDE
(Dept. of Computer Engineering)
DELHI COLLEGE OF ENGINEERING
BAWANA ROAD, DELHI - 110042

ACKNOWLEDGEMENT

This work is not a solo endeavor but rather the amalgamate consequence of contribution from various people and sources. Therefore it would be discourteous to present it without acknowledging their valuable guidance.

It is distinct pleasure to express my deep sense of gratitude and indebtedness to my learned supervisors Dr. Daya Gupta, Mrs. Rama Hariharan (technical director NIC) for their invaluable guidance, encouragement and patient reviews. Their continuous inspiration only has made me complete this dissertation. All of them kept on boosting me time and again for putting an extra ounce of effort to realize this work.

I would also like to take this opportunity to present my sincere regards to my teachers Prof. D.Roy Choudhary, Mrs. Rajni Jindal, Dr. S. K. Saxena and Mr. Manoj Sethi for their support and encouragement.

Finally we are also thankful to my classmates for their unconditional support and motivation during this work. Last but not least, I special thanks to the crowd who are active in the field of Semantic Web.

(ANUJ KUMAR)
Master in Engineering
(Computer Technology & Application)
Dept. of Computer Technology
DELHI COLLEGE OF ENGINEERING
BAWANA ROAD, DELHI - 110042

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

The web, once solely a repository for information, is evolving into a provider of services, such as e-commerce and searching. With this huge amount of information available the ***S**emantic **W**eb **T**echnologies* are one of the promising solutions for efficiently managing huge free form data in web documents. The Semantic Web aims to enrich the existing web with a layer of machine-interpretable metadata so that a computer program can draw conclusions predictably.

Enabling the web with semantic web technologies is one of the promising solutions for efficiently managing huge free form data in web documents. The HTML language has a major drawback. It defines mark-ups that define the format in which a given text should be displayed but says nothing about the text. Thus internet user is presented with lot of irrelevant documents by present search engines, because they simply have no way to tell what is in the documents. More over they have no way to tell what a given word in the document means. These problems are solved by the Semantic Web Technology.

To create semantic web infrastructure based on current web and technologies some of the requirements are: The techniques for developing ontology, mechanism for semiautomatic population of ontology and methods for semantically annotating the documents.

In this thesis we present state of art methods available for developing the ontology, semi automatically ontology population and semantically annotating documents. We go on to build a tool for above services using existing techniques. This tool has been designed as a prototype for the plug-in of ***N**ational **P**anchayat **P**ortal* at ***N**ational **I**nformatics **C**entre* (NIC India) to make it as India's first semantic portal.

1. INTRODUCTION

1.1 Introduction of Semantic Web

In the current web the enormous increase of data has made it difficult to find, access, present and maintain the information required by a wide variety of users. It is facing new problems such as information overload, knowledge representation, population and inefficient searching and organizing data. These problems will be resolved by the modern technology known as ***S**emantic **W**eb **T**echnology* [1][2]. The basic building technologies for knowledge representation are RDF, a language recommended by W3C for semantic web and ontology design for particular domain. Tim Berners-Lee, Director of the World Wide Web Consortium, referred to the future of the current WWW as the "*semantic web*" - an extended web of machine-readable information and automated services that extend far beyond current ***c**apabilities* [3]. To represent knowledge in semantic web an appropriate knowledge representation scheme for any domain is important to represent data in effective way. How to represent and organize data in reusable form to reduce cost and ease of operation on data is the main issue with current web which can be resolved by the concept of ontology, annotation and semantic search with the concept of semantic web.

Today's Web was designed primarily for human interpretation and use. Nevertheless, we are seeing increased automation of Web service interoperation, primarily in search and e-commerce applications. The primary goal of today's search and browsing techniques is to find relevant documents. As the current web evolves into the next generation termed the Semantic Web, the emphasis will shift from finding documents to finding facts, actionable information, and insights.

The information on the internet is in the form of documents. Documents use a mark up language HTML to organize data for display. Mark ups add some information to the text. This additional information helps to organize text in a much better way than plain text. This HTML language has a major drawback. It defines mark-ups that define the format in which a given text should be displayed but says nothing about the text. Thus internet user is presented with lot of irrelevant documents by present search engines, because they simply have no way to tell what is in the

documents. More over they have no way to tell what a given word in the document means. These problems can be solved by modern technology known as ***S**emantic **W**eb*.

Research into the Semantic Web has been continuing for almost a decade. The early days of research concentrated more on the theoretical aspects of ***O**ntologies*[4], description logics and how to formally represent knowledge on the Web. Recently, research has become more concentrated on practical applications that can use the vast amount of knowledge that has been published as *RDF*[5] or OWL.

1.2 Proposed Research Work

To explore this area further we were sent to National Informatics Centre (NIC) by our college. We worked at NIC for its project ***N**ational **P**anchayat **P**ortal* designed and developed using an open eNRICH. The portal is developed for all the Village Panchayats in India. There are more than two lakhs ***V**illage **P**anchayats* in India. It allows people to upload data on the website. After people upload data files, they are examined by the administrator. While uploading the documents the portal asks for some metadata like name of the author, something about the content etc. The administrator then reads the file and puts it in appropriate port-lets to be displayed. The portal also provides a search engine that allows users to find the content on the website**.** In addition to it the portal is designed to be multilingual to support the people speaking and knowing different languages.

We found that construction and application of National Panchayat portal involves a lot of data that is generated by user every day, as the number of Village Panchayats is huge, the amount of data uploaded on the website is also huge. And it's extremely time consuming to do all this manually. An obvious solution is to try and automate this whole process.

Therefore making the use of existing tools and technologies I choose **"Design and Implementation of Semantic Annotation Tool with Ontology Creation and Semi-automatic Ontology Population."** as the topic of research for my thesis work.

1.3 Related Work

A lot of work is being done in the field of semantic web. A number of approaches are being presented. We came across one such paper [10]. This paper presents and alternative way of creating semantic web. It calls such web, A ***S**oft **S**emantic Web*[10]. This paper presents an approach to creating semantic web without using ontology. It proposes to use statistical approach by taking joint frequency of the keywords as the a parameter. It suggests that such joint keyword frequency can represent concepts. Another paper by Rada F. Mihalcea and Silvana I. Mihalcea [11] propose the use of ***W**ordNet*[12] to enhance the information in the documents which can be used to increase the efficiency of information retrieval.

1.4 Proposed System

To automate the process mentioned in forgoing section 1.2, we found that existing tools and techniques were either too complex or not suitable to our project. Therefore we went on to develop ***technique*s** and ***tool*** required for the project. After analysing the problem, we required techniques for following services:

- **Ontology development for portal domain.**
- **Ontology based Annotations.**
- **Semi-Automatic Ontology Population.**

We go on to build a tool for above services. This tool has been designed as a prototype for the plug-in of ***N**ational **P**anchayat **P**ortal* at National Informatics Centre (NIC India) to make it as ***India's first semantic portal.***

1.4.1 Ontology Design

Since data for the National Panchayat Portal is scattered on the web so for this we need an integrated knowledge base. To organise and represent this huge data we need ontology for our

project domain. To develop ***O**ntology*[6] for National Panchayat Portal we analysed the data that is published and uploaded onto it and needed an approach for creation of ontology .We then developed an approach and identified the classes, instances and various relationships that exist between those classes. Since different ontology editors (Tool for creating ontology) for ontology creation are available we use ***p**rotégé*[7] for developing our ontology, as it is easy to use and provide functionality for checking completeness and consistency.

1.4.2 Ontology Based Annotation

Once we have developed our ontology for the portal, for Semantic Annotation we need techniques to automatically read and understand the documents uploaded. To achieve this we perform ***N**atural **L**anguage **P**rocessing*[8] (NLP) on the documents so as to extract information. We used ***G**ATE*[9] as a tool for Information Extraction. The entities identified during NLP are then looked into the ontology and then document is annotated with corresponding classes in ontology. These techniques implemented in tool is called ***A**nnotator* which annotates the document given as input using ***G**ATE*.

1.4.3 Semiautomatic Ontology Population

Since the portal involves the continuous flow of data, so lots of new instances are identified everyday so to add these instances into the ontology we have developed an approach which involves minimum user intervention at administrator end. As the documents are processed by ***A**nnotator*, some new entities are generated and administrator is presented with these entities. The tool prompts ontology class hierarchy to which that instance can be added. Further administrator is asked to verify these types. After verification the instances are added to ontology. In this way we achieved partial automation in Ontology Population.

1.5 Organization of work

The remainder of thesis is organized as follows.

In Section 2 we need to have an overview of the semantic web fundamentals and the basic building blocks for semantic web, RDF/RDFS, Ontologies and its language OWL, Annotation with semantic annotation, URL's and Semantic web architecture, information and noninformation resources, basic idea of HTTP and XML in brief.

In section 3 we covers the introduction semantic search that is the final aim of researchers community working on semantic web with problems that's arises with simple content based search mechanism. It this section we also describes the approaches for creation of Ontology with some well defined steps as it required to organize the data scattered on the web into an integrated knowledge base and semi automatic ontology population mechanism that is required to keep our ontology complete and consistence.

Section 4 deals with the NLP issues that is required to automatically read the documents and identifying the entities for semantic annotation and ontology population. It also covers the introduction to NLP, natural language understanding with natural language text processing systems. It also describes in detail IE (Information Extraction) and Semantic Annotation with definitions.

Section 5 presents the Architecture of the tool (Semantic Annotation Tool with semiautomatic ontology population) with its functional diagram and the design approach I have used for building the tool.

Section 6 covers the implementation details with the block diagram and different classes which I have implemented in JAVA & JSP. It shows how various modules communicate with each other. It also shows the state of art GUI we have developed for the tool using AJAX.

During the period of working over this project we interacted with professionals in the field of semantic web we incorporated their reviews. We communicated our approach for developing semantic web infrastructure with some International conferences; couple of our papers are published. Section 7 describes the papers we have published.

Section 8 cover the conclusion of the work done by us. We finally culminate thesis showing different references including research papers, web sites and books that I had gone through during my project.

Appendix 1, 2, 3 describes GATE, Protégé & Lucene ,the available tool used for ***"Design and Implementation of Semantic Annotation Tool with Ontology Creation and Semi-automatic Ontology Population".***

2. SEMANTIC WEB FUDAMENTALS

This section provides a general introduction to the Semantic Web and its associated technologies. The main components of the Semantic Web including RDF, OWL and URIs are all discussed. An introduction is also given to Ontology, Annotation and Semantic Annotation.

2.1 The Semantic Web

The World Wide Web as it exists today consists of many millions of documents accessible through The Internet. Documents can consist of text, audio, video or indeed can be of any format that can be read on a computer. These documents are mainly produced for human consumption, i.e. interaction by a user or simply text that can be read. Therefore the majority of the documents on the Web are meant to be understood by humans and used by humans. Although the Web has provided a useful and sometimes invaluable resource, there is a potential to make the Web even more useful than it is today. A new level of functionality could be added to the Web if, instead of being human oriented, the web or documents on the web could be understood by machines as well as humans. The need to make The Web machine understandable sparked the beginning of The ***S****emantic* ***W****eb* [1].

The Semantic Web is an effort, to effectively organize all of the knowledge on the Web, Make it machine understandable using a common set of standards, make it universally available to different forms of devices; and provide services that will help us achieve tasks that, at the moment, require some amount of time and effort.

To give us an idea of what The Semantic Web would look like a futuristic scenario was presented in which electronic intelligent agents communicate with each other using the Semantic Web to automatically create doctors appointments and buy medication on prescription (*BernersLee* 2001). There are many such scenarios in which The Semantic Web could fully automate our everyday tasks. When somebody would like to go on holiday abroad, The Semantic Web would enable the flights, travel, accommodation and entertainment to be booked

automatically. When someone would need to replace a part of their computer, they would only need to describe that component and then all of the manufacturers and sellers would automatically be contacted and the best priced component would be ordered. This is obviously a very challenging goal and requires a lot of effort and research in order to make it a reality. The approach so far has been to build each part of the 'cake' (see next section) piece by piece until eventually all the different areas of research converge. During the last few Year's significant progress has been made in laying the foundation of The Semantic Web which has now culminated in W3C approved specifications for knowledge representation (RDF) [5] and ontology definition (OWL). With these specifications in place applications are now being developed that demonstrate the potential power of The Semantic Web (*Shadbolt* et al. 2004), (*Karger* et al. 2005), (*Noy* et al. 2000).

Semantic Web is being developed in order to overcome the following main limitations of the current Web:

- No structured information - the majority of web documents has a lack of structure regarding the representation of information;
- Ambiguity of web content - the inexistence or poor aggregation of information to specific contexts;
- Inadequacy for automatic information transfers;
- Increasing difficulty to locate, access, present and maintain an online trustful content for an increasing number of users;
- Inexistence of universal mechanisms able to provide the capacity to machines to understand the information.

Semantic Web goal is not to make computers understand the human language, but to define an universal model for the information expression and a set of inference rules that machines can easily use in order to process and to relate the information as if they really understand it Though, as the current Web allowed the sharing of documents among previously incompatible computers, the Semantic Web intends to go beyond, allowing stovepipe systems, hardwired computers and

other devices to share contents embedded in different documents. The main advantages of Semantic Web can be grouped into several features as follows:

- Allows the description and spread of the current Web, adding to it a concept layer.
- Allows machine-readable, interpretable and editable web content.
- Offers a way to enable semantic annotations that could be easily organized and found.
- Enhances search mechanisms with the use of Ontologies.
- Enables software agents to carry automatically sophisticated tasks - through the use of smart data.
- Allows better communication between platform independent software agents.
- Enables the use of levels of trust for information.

2.2 The Semantic Web Concepts

2.2.1 RDF/RDFS

The Semantic Web is based around a core set of standards that have been developed by the W3C (***W**orld **W**ide **W**eb **C**onsortium*). This is commonly described as the **S**emantic **W**eb **L**ayer **C**ake', (*BernersLee*, T., 2007) and is shown in Figure 2.1.

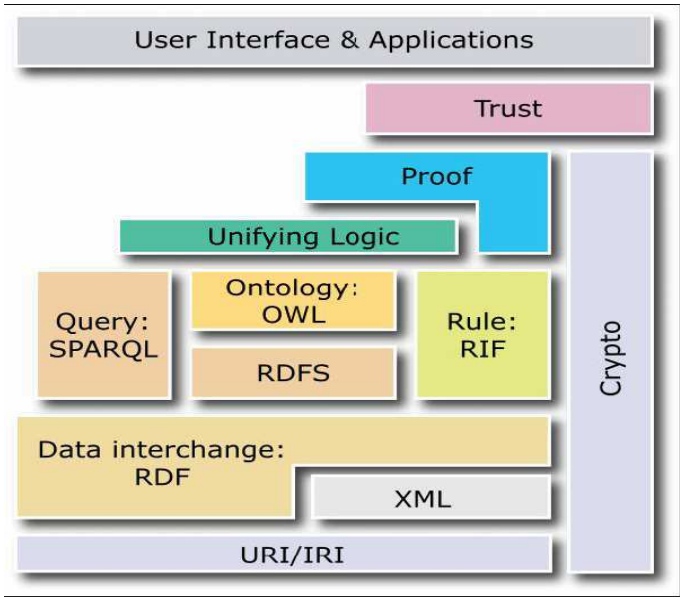


Figure 2.1: Semantic web Architecture

The most important language shown in figure 2.1 is ***RDF*** [5] and its associated vocabulary description framework RDF Schema. The Resource Description Framework is a standard that 'provides interoperability between applications that exchange machine understandable information on the Web' (*Lassila & Swick* 2002). The specification enables things in the real world to be described in the same way that humans use sentences to describe things.

The basic description takes the form:

<subject> <predicate> <object>

A subject takes the form of a URI (Uniform Resource Identifier), which is a way of identifying any object in the world. It looks the same as a URL but the address which is given may not be accessible or may not even exist on the Web, instead it is used as a way of linking objects to some definite identifier. The predicate part, which is also a URI, describes the property of the subject. The object is the thing that is doing the predicate. For example the sentence, 'The Delhi Engineering College has a student called Anuj Kumar', contains a subject, 'The Delhi engineering College', a predicate, 'student' and an object, 'Anuj Kumar'. Such a sentence is easy to translate
into basic RDF/XML syntax and is expressed as:

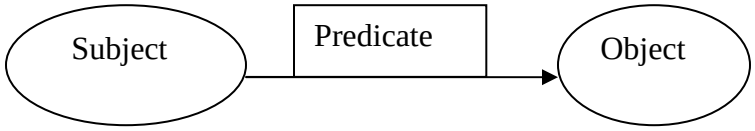


Figure 2.2: RDF/XML syntax representation

<rdf:RDF>
<rdf:Description about="http://www.dce.ac.in">

<akt:hasstudent>Anuj Kumar</akt:hasstudent>
</rdf:Description>
</rdf:RDF>

This assumes that the entity The Delhi Engineering College is represented by the URI, "http://www.dce.ac.in". The '*akt:*' refers to a namespace prefix that is declared in a namespace declaration at the beginning of the RDF document. The declaration contains the identifier where the schema to describe the property 'has student' can be found. RDF also contains many other features such as containers and ways to make statements about statements. The RDF specification can express a number of different properties of entities and relationships between those entities. The RDFS specification builds on RDF to allow the creation of classes and subclasses to be used in much the same way as they are used in taxonomic classifications. Together, RDF and RDFS have become the accepted way of describing knowledge of specific domains and information about real world entities.

2.2.2 Ontologies and OWL

O*ntologies*[4] can play a crucial role in enabling Web-based knowledge processing, sharing, and reuse between applications. Generally defined as shared formal conceptualizations of particular domains, ontologies provide a common understanding of topics that can be communicated between people and application systems.

There are some disputes in knowledge representation as to what exactly '*Ontology*' is We consider '*Ontology*' is built which includes a collection of domain-specific concepts, and it is a system description which should include categories, relationships and constraints between them. Ontology, Thesaurus and Taxonomy are method of the information organization.

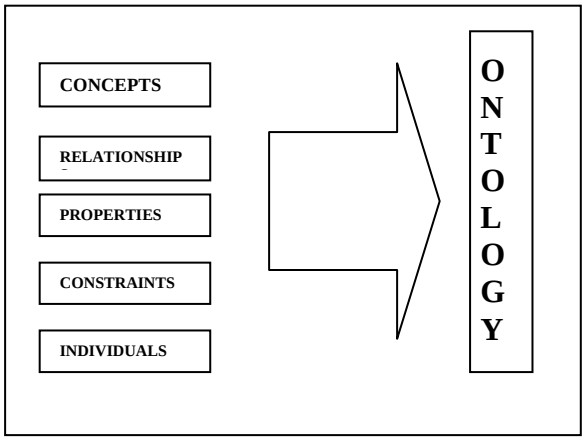


Figure 2.3: Ontology skelton

In the design of ***o****ntology*[6], the basic element is a '***c****ategory*', which meaning its domain. Each category should have '***i****ndividual*' and '***i****nstance*'. Their basic relation is '***i****nheritance*' and '***i****mplement*'. In order to depict them, '***S****lot*' could be used. At the same time, slot is divided into attribute and relationship and limited with facet.

In order for a machine to be able to understand objects in the real world, there needs to be a way to represent and classify these objects. We define and associate objects in the real world by the knowledge that we have about them. Ontology is a means of being able to store knowledge about a particular subject area or for multiple subjects within some specific domain, and process it in some way.

These descriptions are rather vague and there is no way of showing how an ontology can be represented in a form that a computer can read and process. Previous research has led to the development of languages to represent ontologies such as description logics, Frame Logic (*Kifer* et al. 1995) and Knowledge Interchange Format (KIF) (*Genesereth* et al. 1992). There have also been full scale applications made from these languages that are in use today (*Fensel,* 2001).

The main use of ontology is to describe a domain using an appropriate vocabulary and define relationships between the terms taken from the vocabulary. One of the most natural ways for humans to classify objects is to use hierarchies. The ***R****DFS* [5] specification reflects this by using

classes and subclasses and has the ability to define properties that can be given to an instance of a class. However, in order to establish more complex relations, such as cardinality constraints, optional constraints or disjointness, a more complex syntax is required. This has led to the development of OWL (Web Ontology Language) which has now become the standard for ontology definition. This language defines syntax for describing classes and properties as well as more complex relationships. The OWL language 'is designed for use by applications that need to process the content of information instead of just presenting information to humans' (*McGuiness & Harmelen* 2004).

There are three types or 'flavours' of ***OWL***[13]: OWL Lite, OWL DL and OWL Full. OWL Lite is a subset of OWL DL and OWL DL is a subset of OWL Full. The main difference between OWL full and the other two flavors of OWL is that in OWL Full instances of a class can be classes, where as in OWL Lite and OWL DL, instances of a class must be individuals. In practice, this means that working with OWL Full is generally too complex for a logic reasoner to use for logical deduction, but OWL DL is both complete and decidable and is therefore easier to reason over and use.

2.2.3 Annotation

Annotation', in contemporary English, according to ***WordNet*** [12], have two meanings:

1. Note, annotation, and notation: a comment (usually added to a text).
2. Annotation, annotating -- the act of adding notes.

In linguistics (and particularly in computational linguistics) an annotation is considered a formal note added to a specific part of the text. There are number of alternative approaches regarding the organization, structuring, and preservation of annotations. For instance, all the markup languages (HTML, SGML, XML, etc.) can be considered schemata for embedded or in-line annotation. On the contrary, open hypermedia systems use stand-off annotation models where annotations are kept detached, i.e. non-embedded in the content.

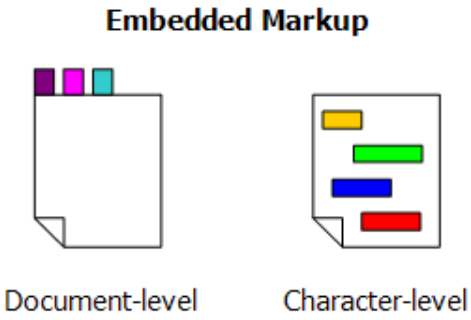


Figure 2.4: Embeded Annotion

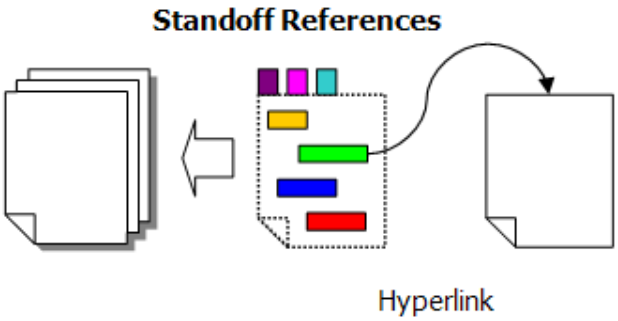


Figure 2.5: Standoff Annotation

We annotate data all the time: when we read a paragraph, and mark “great!” in the margin, that is an annotation. When our text editor underlines a misspelled word, which is also an annotation. Annotations add some information to some other information; to annotate means “to make notes or comments”.

Another way to view annotations is metaphorically: URIs are the “atoms” of the Semantic Web and semantic annotations are the “molecules”. The Semantic Web is about shared terminology, achieved through consistent use of URIs. Annotations create a relationship between URIs and build up a network of data.

2.2.4 Semantic Annotation

***S**emantically **A**nnotated* [14] documents play an important role in semantic web by using and inserting tags according to the meaning of the word in context. The semantic meanings and interpretations of keywords are bound to that domain and therefore the retrieval is likely to be more efficient.

2.2.5 URLs and Semantic Web Architecture

The base of the layer cake shown in figure is URIs and IRIs. Uniform Resource Identifiers and their corresponding Internationalized Resource Identifiers are central to the design and use of the Semantic Web. A URI provides a common syntax that can be used to name or identify any entity in the world.

For example, the URI “http://id.dce.ac.in/person/6751” is an identifier used to refer to the person ‘Anuj Kumar’ that has been provided by DCE. This URI uses the http naming scheme although other schemes are available such as urn, dav, file and ftp. The universality of the URI is a fundamental part of Web architecture. The idea of using URIs is that they should be something into which any system’s identifiers can be mapped. Providing a name for different entities and resources means that descriptions of objects can be easily made and published so that other agents or systems can combine knowledge from different sources. This relies on the assumption that every object will have a unique URI. However, anyone who wants to make statements or give knowledge about an object can introduce their own URI to refer to that object. Thus, there maybe many URIs that refer to the same real world entity. The relationship between a URI and a URL is a subtle one. In the early days of the Web it was thought that the URL was a subclass or special type of URI (W3C, 2001). The contemporary view is that a URL is only an informal concept that provides the address of a location to access a resource. Therefore it can be said that a URI that uses the ‘http’ scheme is also a URL.

2.2.6 Information and Non-Information Resources

The current Semantic Web architecture divides entities and objects that have URIs into two categories: information resources and non-information resources. Non-Information resources are those entities that cannot be represented as a byte stream or serialized into a character format. This category of objects covers things such as cars, people, places, books and concepts that exist in the real world and not on the Web. The category of information resources on the other hand, consist of resources that can be accessed on the Web such as HTML pages, JPEG images, PDF documents and RDF descriptions. Information resources can be used to describe non-information resources in a way that can be accessed using HTTP. The two categories of resource have been made in order to be able to represent real world entities on the Web. The URI of an object cannot also contain the description of that object.

In order to use URIs and make statements about the objects that the URIs refers to, a mechanism had to be constructed so that the URI and description stay separated. The way that this is

accomplished is to introduce an HTTP 303 redirection when the URI of a resource is resolved or dereferenced on the Web. For example, when the URI for 'Anuj Kuamar ' is put into a Web browser, the browser will issue an HTTP GET to the server, who in turn will redirect the browser, via 303, to a description of the URI in the format that was requested by the browser. Each description of the URI will itself have a URI so that the distinction between information and noninformation resources is preserved. The architecture is illustrated in Figure with the URI for 'Anuj Kumar' used together with the URIs for a text/html and rdf+xml description of the URI.

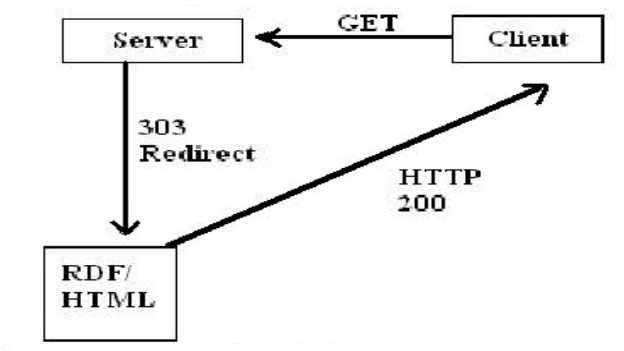


Figure 2.6: Semantic Web architecture for the resolution of URIs of non-information resources

2.2.7 Introduction to HTTP

The ***H**ypertext **T**ransfer **P**rotocol* (HTTP) is a generic and stateless protocol for distributed, collaborative, hypermedia information systems. It allows performing operations on resources which are identified by URIs. It has been widely used for more than one decade and now comes to version 1.1. It has four main methods:

a. Get means get the information that is identified by the request URI. Usually it is the action we take when browsing sites and clicking hyperlinks, we ask the web server for the resources that is identified by Request-URLs.

b. Post means make a request to the web server so that the server accepts the resources encapsulated in the request, which will be the new subordinate of the resource identified by the Request-URI in the Request-Line. Usually it is what we do when filling and sending the HTML form to the server to buy something like CDs, books etc. -- making a request of the server.

c. Put means make a request that sends updated information about a resource if the resource identified by the Request-URI exists, otherwise the URI will be regarded as a new resource. The main difference between the POST and PUT requests lies in the different meaning of the Request-URI. In a POST request, the URI is to identify the resource that will handle the enclosed entity. As for the PUT request, the user agent knows what URI is its aim and the web server cannot redirect the request to other resources. Unfortunately most web browsers don't implement this functionality, which makes the Web, to some extent, a one-way medium.

d. Head is similar to GET except that the server don't return a message-body in the response. The benefit of this method is that we can get meta-information about the entity implied by the request without transferring the entity-body itself. We can use this method to check if hypertext links are valid, or if the content is modified recently .By using HTTP, Semantic Web can benefit all these functionalities for free. In addition, almost all HTTP servers and clients support all these features.

2.2.8 XML

Extensible Markup Language (XML) is a subset of SGML (the Standard Generalized Markup Language), i.e. it is totally compatible with SGML. But it is simple and flexible. It's original aim to tackle the problems of large-scale electronic publishing. However, it is also very important in data exchange on the Web. Despite its name, XML is not a markup language but a set of rules to build markup languages.

Markup language

"Markup is information added to a document that enhances its meaning in certain ways, in that it identifies the parts and how they relate to each other." - Erik T. Ray, Markup language is kind of mechanism organizing the document with a set of symbols, e.g. this article is labeled with different fonts for headings. Markup use similar methods to achieve its aims. Markup is important to implement machine-readable documents since a program need to treat different part of a document individually.

Namespaces

We can expand our vocabulary by namespaces which are groups of element and attribute names. Suppose, if you want to include a symbol encoded in another markup language in an XML document, you can declare the namespace that the symbol belongs to. In addition, we can avoid the situation that two XML objects in different namespaces with the same name have different meaning by the feature of namespaces. The solution is to assign a prefix that indicates which namespace each element or attribute comes from. The syntax is shown below:

ns-prefix*: *local-name

XML Schemas

XML itself does not do anything, i.e., it is just structure and store information. But if we need a program to process the XML document, there must be some constraints on sequence of tags, nesting of tags, required elements and attributes, data types for elements and attributes, default and fix values for elements and attributes and so on. XML Schema is an XML based alternative to Document Type Definition (DTD) . There are some features of XML Schemas that overweigh DTD:

a.) XML Schemas support data types, which brings a lot of benefits, e.g. easy to validate the correctness of data, easy to work with databases, easy to convert data between different types.
b.) XML Schemas have the same syntax as XML so that it can benefit all features of XML.
c.) XML Schemas secure data communication since it can describe the data in a machine-understandable way.
d.) XML Schemas are extensible because they are actually XML and then share this feature of XML.
e.) Well-formed is not enough since it also may contain some semantic confusion which can be caught by XML Schemas.

3. SEMANTIC SEARCH AND ONTOLOGY DESIGN

This section will explain in detail about semantic search and the problems that arises with simple content based search it also compares the results that are that are generated during searching with simple search engines. This section also describes in detail Ontology design approach that we have gone through during designing our ontology for National Panchayat Portal.

3.1 Semantic Search

Current statistics show that the number of Web pages is in excess of one billion (*Madhavan* et al., 2007). With such a huge amount of data available some means had to be put in place so that exact information, relating to a particular purpose, could be searched for and retrieved. This need for relevant information spawned the production of search engines dedicated to sift through the Web and pick out documents and content that satisfy the requirements given to the engine by a user. However, a successful search for information can only be as good as the ability of the search engine to retrieve that information. Web pages, and any form of information given as text, is only read by a machine but is not understood by a machine. This means that although a text search for a word will be able to find that particular word, the machine cannot attach any meaning to the words it finds or is looking for. This severely limits the ability of a machine to retrieve data according to the needs of a human.

For example, if a person wanted to search a document or group of documents for information about ‘men’s clothes’ they would have to tailor their thinking to extract the words that they thought an article that talked about this subject would contain, such as shirts, ties, coats, sizes, materials, price and so on. If, however, the machine somehow knew that a shirt was an item of clothing, or that coats come in four sizes and knew that ties could be of silk or cotton then the machine could automatically give all this information without the need for multiple searches.

3.2 Problems of Search and Retrieval on the Web

One of the major uses of the ***W**orld **W**ide **W**eb* today is to use search engines to find out about information. Due to the increase in the number of Web pages available, finding the desired information can often be an arduous and complex task. There are a number of problems with the current approach to finding results on the web:

- ***The most basic problem is that the information that you are looking for cannot be found. This could happen for a number of reasons**:*

1. The information simply does not exist on the web.
2. The search engine could not find the information that does exist.
3. The information is in a web page that is not on the first page of returned results and the user does not try and look on through the next few results pages.

***P**assin* gives an example where he could not find a replacement part for an old stove, even though there was a shop with a website that sold the part.

- ***There may be too many irrelevant results returned.***

The ratio of the number of relevant results retrieved to the total number of irrelevant and relevant results retrieved is called the *precision* of a search. As a brief example, if a person wanted to know about the Isle of Wight Mosque, then entering this search term into Google produces a total number of 38600 results. However, out of these there are only 2 results that are about the Mosque. This gives a precision value of 0.00005%. The other results are pages that contain the words 'Isle of Wight' and the word 'Mosque' with little or no connection between them.

- ***Only some of the relevant results to the query are returned.***

The ratio of the number of relevant results retrieved to the total number of relevant results indexed by the search engine is called the *recall* of a search. This value is harder to measure as it is virtually impossible to calculate how many relevant results to a search exist on the web.

However, *Clarke and Willett* (1997) produce a relative estimation of recall based on pooling the results of a number of search engines to the same query. *Shafi and Rather* (2005) use this method to compare the precision and recall of five search engines in the retrieval of scholarly information in the field of
biotechnology.

- ***The search engine searches for the query term in the wrong context and returns results in that context.***

There are many cases where words have more than one meaning. If a user does not correctly add extra keywords to their search to contextualise the topic, they could receive anomalous results. For example, if somebody wanted to know about the Greek deity 'Nike', simply entering this word into a search engine will return results about the sportswear company Nike.
At present, Google has 46.3% (*Sullivan* 2006) of the search engine market. The reason for Google's success is claimed to be down to the ***P****ageRank* ***a****lgorithm* (*Brin* et al. 1998) which ranks pages, not just by keyword frequency, but also according to how many pages link to a site, how many hits a site has and how often a site is updated, as well as other varying criteria (Arnold 2005). However, statistics also show that other search engines return results that are not found by Google (*Notess* 2002). This all means that the end user cannot make best use of the Web since the information or knowledge that the user requires is either difficult to find or search engine dependent.

The lack of a suitable search engine for the web stems from the fact that all search engines rely heavily on keyword frequency, and in essence, string matching to find their results. When a person enters the term 'football' into a search engine it has no knowledge about the concept of football, it will base its results on the number of times this keyword is mentioned in a document. There are algorithms that enhance this approach somewhat but the underlying principle is the same (*Langville & Meyer* 2004).

What is needed is a system that knows that football is a type of sport and that it has a World Cup and that it is played by 11 players etc. This kind of knowledge awareness is exactly what The

Semantic Web is trying to achieve. If there was ontology for football then all of the above facts could be easily modeled and used to enhance query results. Then, if someone was to make a query about football, the system would try and attain the concept of the search such as, is the person looking for football tickets? Are they looking for football kit? Or are they interested in a particular team? And so on.

The Semantic Web promises a new generation of World Wide Web infrastructure that will make it possible for machines to 'understand' the data on the web instead of merely presenting it. In order to encourage the increase of semantic web technologies there have been suggestions that a 'killer app' may be needed to convince those that are still unsure about the benefits that semantic web technologies can bring.

The search engine is an example of a potential 'killer app' that has been responsible for increased usage of the current web. As described in this section there are a number of problems with searching the Web today. Despite the improving quality of modern search engines, statistics show that only 17% of people find exactly the information they were looking for (Fallows, 2005). Furthermore, a study has shown that the recall of some search engines can be as low as 18% (*Shafi & Rather*, 2005). There is therefore a need to improve the quality of search results and user experience. The Semantic Web provides an opportunity to achieve such a goal. The use of RDF and OWL as knowledge representation formats can provide structured content to describe a given domain or set of domains. Using this knowledge, it should be possible to add a sense of 'understanding' to a search engine when searching for results whose knowledge has been partly or fully described in a knowledge representation format.

In the past, such a proposal may not have been viable due to the lack of ontologies and RDF resources available on the web. However, at the present time there are estimated to be more than 5 million RDF or OWL documents available on the Web (*Ding*, 2006). Even if most of those documents contain knowledge about a limited set of concepts, RDF data from sources such as ***D****Bpedia* and ***W****ordnet* provide a suitable base from which to begin exploring the enhancement that can be made to ordinary Web searches. More importantly, there will be a number of

knowledge bases that will be used to store RDF instance data and OWL ontologies that have the ability of being

3.3 Ontology Design Approach

***O**ntology* [13][15] is a common set of terms that are used to describe and represent knowledge in an organized way. In recent years, semantic web has enriched itself with the help of ontology design in different domains like medical, tourism and knowledge representation in an efficient way. There are many knowledge representation systems developed using ontology for semantic web development. In the forgoing domains, the design of ontology has been done is an Ad-hoc way. There are many problems during the ontology design process between concepts and to become real for particular domain. Different approaches have been used for building ontologies, most of them to date using mainly manual methods.

In the design of ***o**ntology* [16][17] the Basic elements are concepts, relationship, constraints, individual, and properties which will define ontology for a domain.

- **CONCEPTS** are the classes that are needed to be identified for the particular domain.
- **RELATIONSHIP** defines of individuals of different classes are related to each other.
- **PRPERTIES** are the attributes of individuals of the concepts identified for a domain.
- **CONSTRAINTS** are the conditions that are needed to be satisfied during ontology design.
- **INDIVIDUALS** are the entities of the class or concepts.

There have been some methodologies for building ontologies developed, again assuming a manual approach. For instance, the methodology proposed in (***U**schold and **K**ing*) involves the following stages:

1. ***Identifying the purpose of the ontology (why to build it, how will it be used, the range of the users).***

This step ontology design process involves analysis of the domain for which we are going to design ontology. Here we identify the goal and scope of the knowledgebase to be created.

2. ***Building the ontology.*** It is further divided into three steps.
   - **i)** The first is ontology capture, where key concepts and relationships are identified, a precise textual definition of them is written, terms to be used to refer to the concepts and relations are identified, the involved actors agree on the definitions and terms.
   - **ii)** The second step involves coding of the ontology to represent the defined conceptualization in some formal language (committing to some meta-ontology, choosing a representation language and coding).Evaluation and documentation.
   - **iii)** The third step involves possible integration with existing ontologies.

3. ***Evaluation and documentation.***

Once the ontology is created we have to perform validation check for the ontology to check the quality and knowledge representation efficiency of the ontology. If there is constraints violation during the design then we will remove it using design phase.

3.4 Semiautomatic Ontology Population

Since web or portal is continuously flooded by user's data, it is therefore required to populate the knowledgebase (ontology) created to make it complete and consistent with the changing requirements. Manual population of ontology is a time consuming and inefficient approach, so it some automatic mechanism is required for this, we call it as ***S****emiautomatic* ***O****ntology* ***P****opulation*. The approach to semi automatically populate the ontology is described in detail in the design section.

Since the tool we have developed is for a web portal so some online support for viewing, population and modification should be provided we have also integrated a web interface of protégé[18] for this ,so that administrator can use this feature .

Tool and Language:

We have used ***protégé 3.3.1***[7] as an ontology editor (also as knowledge representation tool) with OWL as the underlying representation language. Protégé 3.3.1 is a frame based development and management system that offers classes, properties, forms and individuals as the building blocks for representing knowledge.

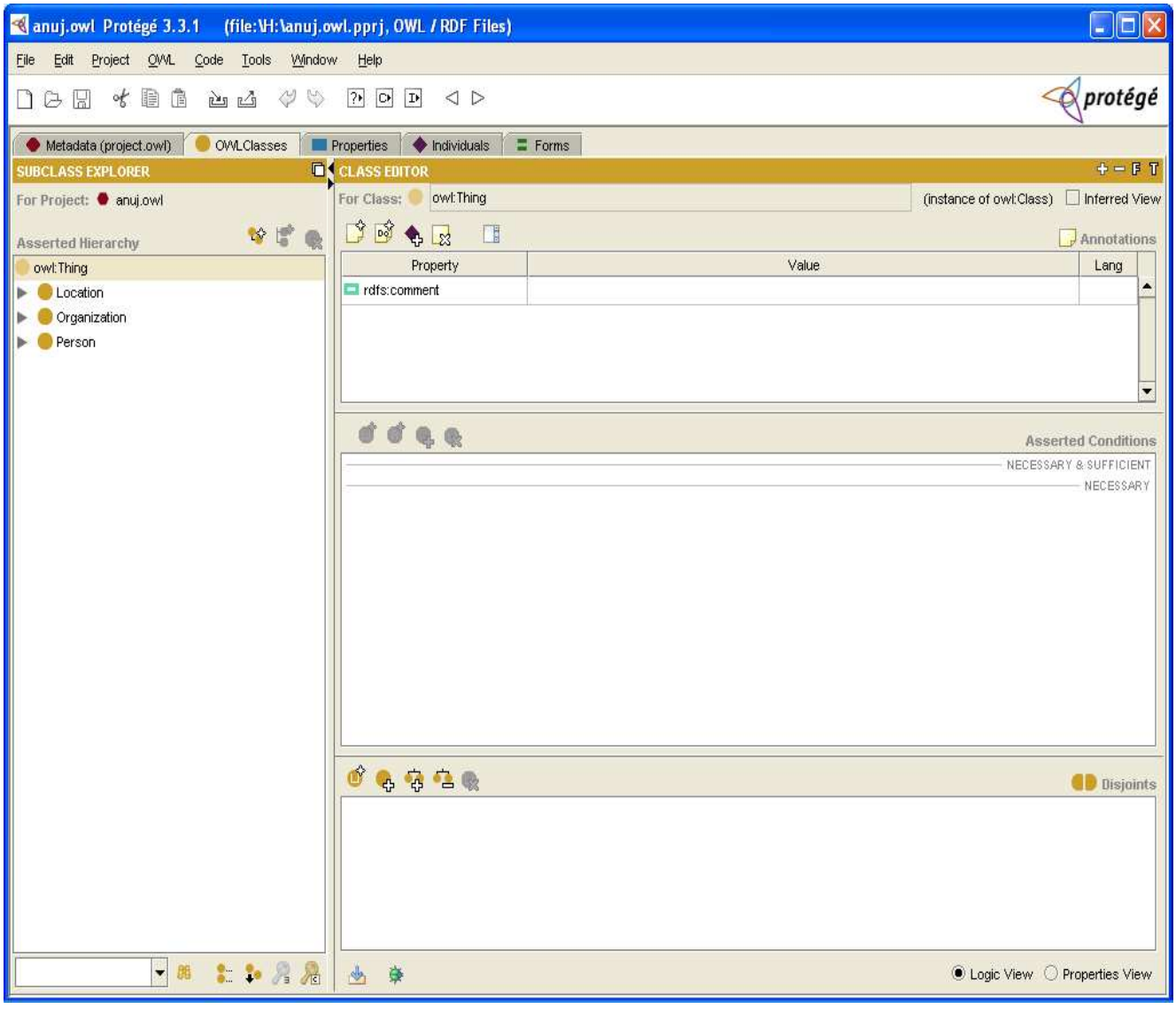


Figure 3.1: Protégé Interface showing Ontology

4. NLP AND SEMANTIC ANNOTATION

4.1 Introduction to NLP

***N**atural **l**anguage **p**rocessing* [8] provides a potential means of gaining access to the information inherent in the large amount of text made available through the Internet. In today web environment where terabytes of data generated every day, it is hard to read all the data generated everyday manually. So, for this some kind of natural language processing is necessary for computer to process this data.

Natural Language Processing (NLP) is an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to do useful things. NLP[8] researchers aim to gather knowledge on how human beings understand and use language so that appropriate tools and techniques can be developed to make computer systems understand and manipulate natural languages to perform the desired tasks. The foundations of NLP lie in a number of disciplines, viz. computer and information sciences, linguistics, mathematics, electrical and electronic engineering, artificial intelligence and robotics, psychology, etc. Applications of NLP include a number of fields of studies, such as machine translation, natural language text processing and summarization, user interfaces, multilingual and cross language information retrieval (CLIR), speech recognition, artificial intelligence and expert systems, and so on.

Beginning with the basic issues of NLP, this chapter aims to chart the major research activities in this area.

1. Natural language text processing systems – text summarization, information extraction, information retrieval, etc., including domain-specific applications.
2. Natural language interfaces.
3. NLP in the context of www and digital libraries.

4.2 Natural Language Understanding

At the core of any NLP task there is the important issue of natural language understanding. The process of building computer programs that understand natural language involves three major problems: the first one relates to the thought process, the second one to the representation and meaning of the *linguistic input*, and the third one to the world knowledge. Thus, an NLP system may begin at the word level – to determine the morphological structure, nature (such as part-of-speech, meaning) etc. of the word – and then may move on to the sentence level – to determine the word order, grammar, meaning of the entire sentence, etc.— and then to the context and the overall environment or domain. A given word or a sentence may have a specific meaning or connotation in a given context or domain, and may be related to many other words and/or sentences in the given context.

Liddy (1998) and ***F****eldman* (1999) suggest that in order to understand natural languages, it is important to be able to distinguish among the following seven interdependent levels, that people use to extract meaning from text or spoken languages:

- phonetic or phonological level that deals with pronunciation
- morphological level that deals with the smallest parts of words, that carry a meaning, and suffixes and prefixes
- lexical level that deals with lexical meaning of words and parts of speech analyses
- syntactic level that deals with grammar and structure of sentences
- semantic level that deals with the meaning of words and sentences
- discourse level that deals with the structure of different kinds of text using document structures and
- Pragmatic level that deals with the knowledge that comes from the outside world, i.e., from outside the contents of the document.

A natural language processing system may involve all or some of these levels of analysis.

4.3 Natural Language Text Processing Systems

Manipulation of texts for knowledge extraction, for automatic indexing and abstracting, or for producing text in a desired format, has been recognized as an important area of research in ***NLP*** [8]. This is broadly classified as the area of natural language text processing that allows structuring of large bodies of textual information with a view to retrieving particular information or to deriving knowledge structures that may be used for a specific purpose. Automatic text processing systems generally take some form of text input and transform it into an output of some different form. The central task for natural language text processing systems is the translation of potentially ambiguous natural language queries and texts into unambiguous internal representations on which matching and retrieval can take place. A natural language text processing system may begin with morphological analyses. Stemming of terms, in both the queries and documents, is done in order to get the morphological variants of the words involved. The lexical and syntactic processing involve the utilization of lexicons for determining the characteristics of the words, recognition of their parts-of-speech, determining the words and phrases, and for parsing of the sentences.

4.4 Information Extraction

Knowledge discovery and data mining have become important areas of research over the past few years. ***I****nformation* ***E****xtraction*[8] (IE) is a subset of knowledge discovery and data mining research that aims to extract useful bits of textual information from natural language texts (*Gaizauskas & Wilks*, 1998). A variety of information extraction (IE) techniques are used and the extracted information can be used for a number of purposes, for example to prepare a summary of texts, to populate databases, fill-in slots in frames, identify keywords and phrase for information retrieval, and so on.

N*oun* ***p****hrasing* is considered to be an important NLP technique used in information retrieval. One of the major goals of noun phrasing research is to investigate the possibility of combining traditional keyword and syntactic approaches with semantic approaches to text processing in order to improve the quality of information retrieval. We use here this feature of NLP to populate our ontology and for semantically annotating of web documents. The nouns and proper nouns

identified by using the noun phrasing technique are the candidate entities for populating the ontology and for annotating the documents. The complete approach for annotating and ontology population is described in the section given below.

Manual annotation is difficult, time consuming and expensive. Convincing millions of users to annotate documents for the Semantic Web is difficult and requires a world-wide action of uncertain outcome .For our tool to perform automatic semantic annotation in the documents we have used ***G****ATE*[9] **General Architecture for Text Engineering** or **GATE** is a Java software toolkit originally developed at the ***U****niversity of* ***S****heffield* since 1995 and now used worldwide by a wide community of scientists, companies, teachers and students for all sorts of natural language processing tasks, including information extraction in many languages.

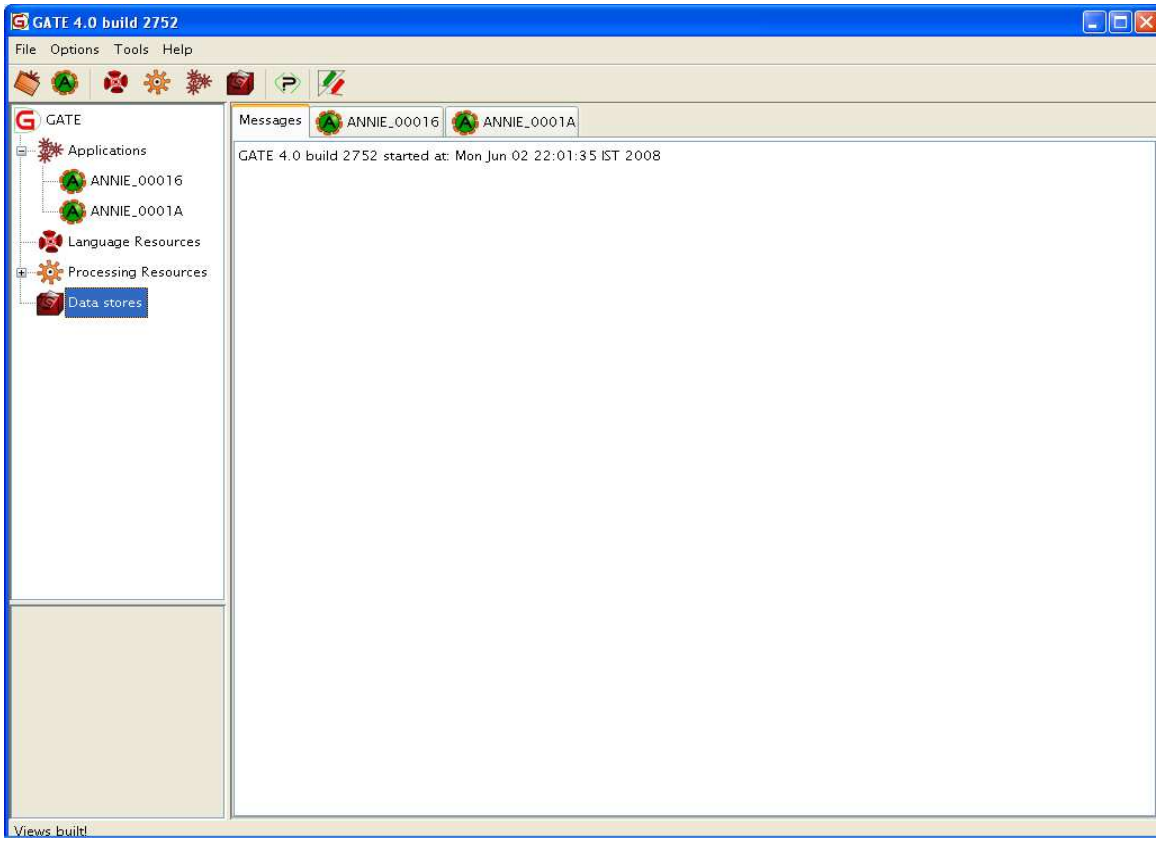


Figure 4.1: GATE

4.5 Semantic Annotation

Annotation as described above in chapter 2 are essential for semantic web, below we describe the conceptual model for it. In this thesis we propose a technique to automatically annotate domain-specific information from large repositories (e.g. Web sites) with minimum user intervention. The complete process of ***s**emantically **a**nnotating* [19] documents is described in coming sections.

Conceptual Model:

We now explore the conceptual model behind annotation. The term "annotation" can denote both the process of annotating and the result of that process. Where we say "annotation" we mean the result. An annotation attaches some data to some other data. An annotation establishes, within some context, a (typed) relation between the annotated data and the annotating data. Investigating the nature of annotation further, we can model it as a quadruple:

Definition 1 (Annotation). *An annotation A is a tuple (a_s, a_p, a_o, a_c), where a_s is the subject of the annotation, the annotated data, a_o is the object of the annotation, the annotating data, a_p is the predicate, the annotation relation, that defines the type of relationship between a_s and a_o, and a_c is the context in which the annotation is made.*

The annotation subject can be formal or informal. For example, when we put a note in the margin of a paragraph, the informal convention is that the note applies to the paragraph, but that pointer is not formally defined. If we however use a formal pointer such as a URI to point to the paragraph then the subject is formally specified.

The annotation predicate can be formal or informal. For example, when we put a note in the margin, the relation is not formally defined, but we may informally derive from the context that that the note is a comment, a change-request, an approval or disapproval, etc. If we use a formal pointer to an ontological term that indicates the relation (e.g. dc:comment) then the predicate is formally defined.

The annotation object can be formal or informal. If an object is formal we can distinguish different levels of formality: textual, structural, or ontological. For example, then string “This is great!” is a textual object. A budget calculation table in the margin of a project proposal is a structural object. And an annotation objects that is not only explicitly structured but also uses ontological terms as an ontological object.

The annotation context can be formal or informal. Context can could indicate when the annotation was made and by whom (provenance), or within what scope the annotation is deemed valid, for example in a temporal scope (it is only valid in 2006) or in a spatial scope (it is only valid inWestern Europe). Usually context is given informally and implicitly. If we use a formal pointer such as a URI then the context is formally defined.

Combining the levels of annotation subject, predicate, and object, we can distinguish three layers in annotations: i) informal annotations , ii) formal annotations (that have formally defined constituents and are thus machine-readable), and iii) semantic annotations (that have formally defined constituents and use only ontological terms).

Definition 2 (Formal annotation). *A formal annotation A_f is an annotation A, where the subject a_s is a URI, the predicate a_p is a URI, the object a_o is a URI or a formal literal, and the context a_c is a URI.*

Definition 3 (Semantic annotation). *A semantic annotation A_s is a formal annotation A_f , where the predicate a_p and the context a_c is an ontological term, and the object a_o conforms to an ontological definition of a_p.*

Semantic Annotation [19] is about assigning to the entities in the text links to their semantic descriptions (as presented on Fig. 1). This sort of metadata provides both class and instance information about the entities. It is a matter of terminology whether these annotations should be called “semantic”, “entity” or some other way. To the best of our knowledge there is no well-established term for this task; neither there is a well-established meaning for “semantic

annotation". What is more important, the automatic semantic annotations enable many new applications: highlighting, indexing and retrieval, categorization, generation of more advanced metadata, smooth traversal between unstructured text and available relevant knowledge. Semantic annotation is applicable for any sort of text – web pages, regular (non-web) documents, text fields in databases, etc. Further, knowledge acquisition can be performed based on extraction of more complex dependencies – analysis of relationships between entities, event and situation descriptions, etc.

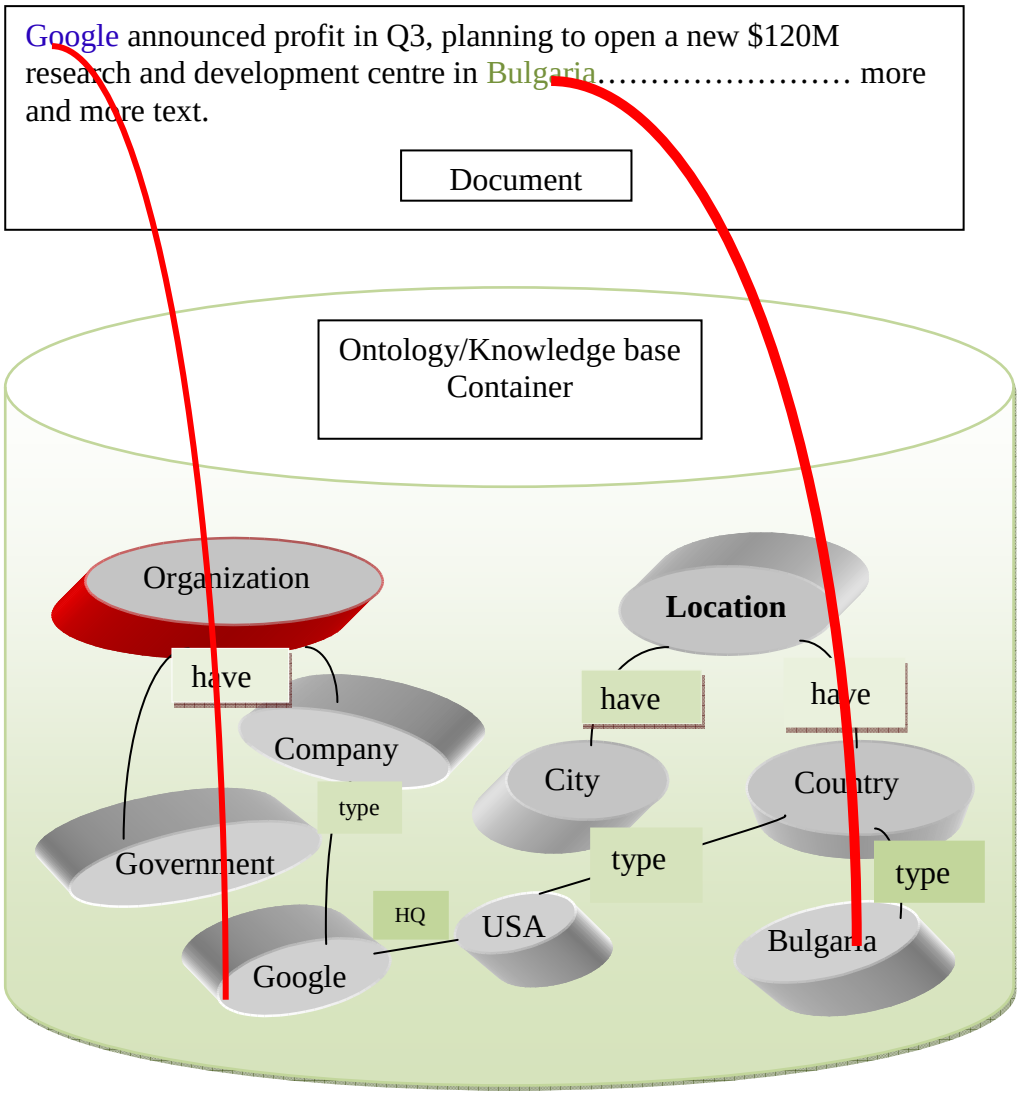


Figure 4.2: Semantic Annotation

5. TOOL ARCHITECTURE AND DESIGN APPROACH

5.1 Tool Architecture

This section describes the overall ***Architecture*** of the tool [20]. The functional diagram for the system is shown in Figure 1. It consists of several components. There are four data stores and five processes. The four data sources are ***O****ntology*, ***D****ocument* ***C****ollection* and two indexes ***D****ocument* ***I****ndex* and ***I****nstance* ***I****ndex*. The Five processes are ***A****nnotator*, ***E****valuator*, ***I****ndexer*, ***S****ummarizer* and finally the ***S****earcher*. Except searcher all other processes are offline processes. There collective job is to transform a document collection into a pair of index table that can be later on searched by the searcher to present the user with the information relevant to the user query.

These processes operate in a sequence. First the annotator annotates the document that is it embeds the *semantic* [14] i.e. meaning related information in the documents using the ontology. Once the documents are annotated these are analyzed by evaluator and categorized and stored in different clusters/groups. Now indexer prepares the index of these annotated documents taking the advantage of both annotated documents and the ontology. Indexer also indexes the summary of the documents; this summary is produced by the summarizer. Finally when the user fires a query searcher searches the index and ontology extract as relevant information as possible for user.

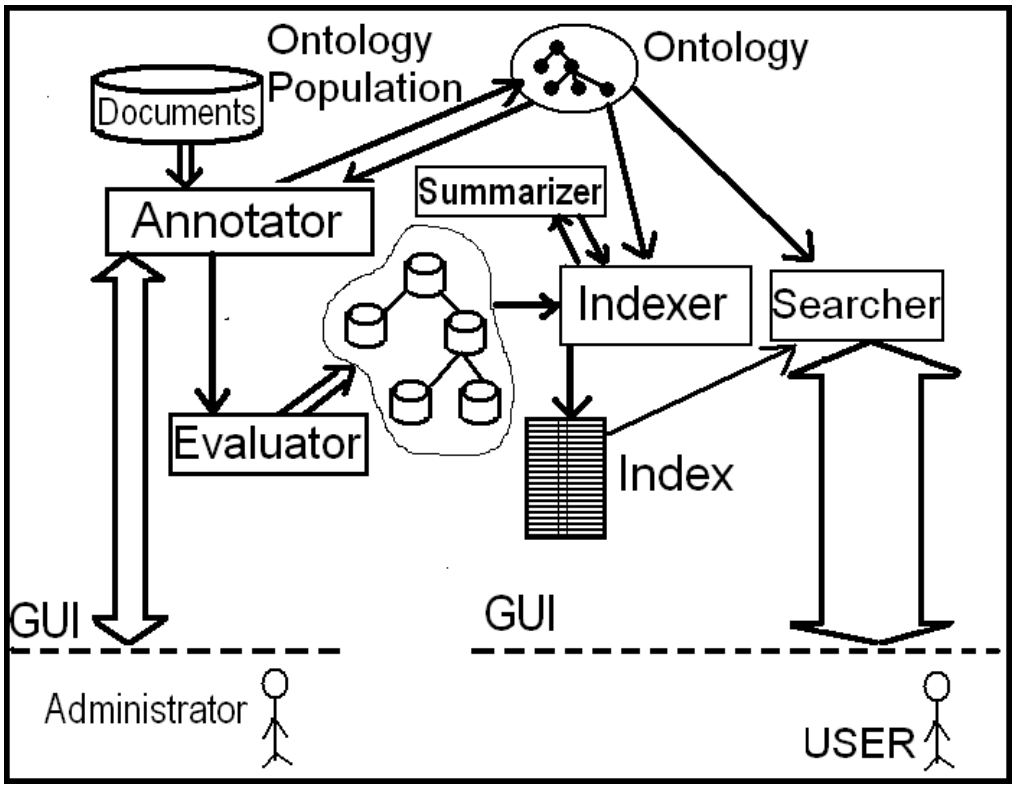


Figure 5.1: Tool Architecture

Role of Administrator: Administrator is person or a team of personnel employed by the organization. The Administrator interacts with the annotator, Indexer, Summarizer and Ontology Population. The Administrators job is to annotate the documents that are uploaded by the users on the web portal using Annotator. Once they are annotated the administrator runs the indexer which automatically prepares the index. The Administrator is also responsible for keep the ontology up-to-date and consistent. He runs the ontology population tool to add new instances to the ontology. Once the new instances are added then administrator has to manually update the relations present in the ontology.

Role of User: The User consists of the people who visit the web portal. User uploads the document on the portal. They also retrieve information from the portal using the search engine.

Here in this thesis I am going to explain with design approach & implementation details the following sections of the tool.

- **Ontology Design for the web portal.**
- **Semantic Annotation.**
- **Semi Automatic Ontology Population.**

Here we also present the block diagram with implementation of the tool.

5.2 Design Approach

5.2.1 Ontology

The current approaches used for data management on the web, can't satisfy the need of efficient data handling and search mechanism for this exponentially increasing data .The data for the National Panchyat portal is spread across different data source so there is need to organize and represent it in efficient way. The organization and management of data is the main issue for knowledge representation. So to represent this data in well organise manner and provide efficient searching within this data it is essential to ***B**uild **O**ntology* [21] for this data.

Through a series of research, we find out that Construction and application of National Panchyat portal *Knowledgebase* is a long term complicated process as this portal involves the lot of data that is generated by user every day. So, for this I had developed a base ontology which can latter be evolved as new concepts are recognized. This ontology evolution is an iterative and never ending process as new concepts are being created everyday. Populating such a huge ontology is also a big problem therefore, we populate using semiautomatic ontology populating mechanism that we have created and will be described later in this thesis.

We use ***P**rotégé 3.3.1*[7] as an ontology editor (also as knowledge representation tool) with OWL as the underlying representation language. Protégé 3.3.1 is a frame based development and management system that offers classes, properties, forms and individuals as the building blocks for representing knowledge.

The ontology for National Panchayat Portal which we have developed since includes the relationship between concepts, and includes the relationship of superclass/subclass/instance etc.

We have developed our base ontology using Protégé 3.3.1[7] which contains 36 classes (concepts, Common nouns, types etc.), properties, constraints and individuals for defined concepts shown in figure below, there are three top level classes Location, Organization, and Person.

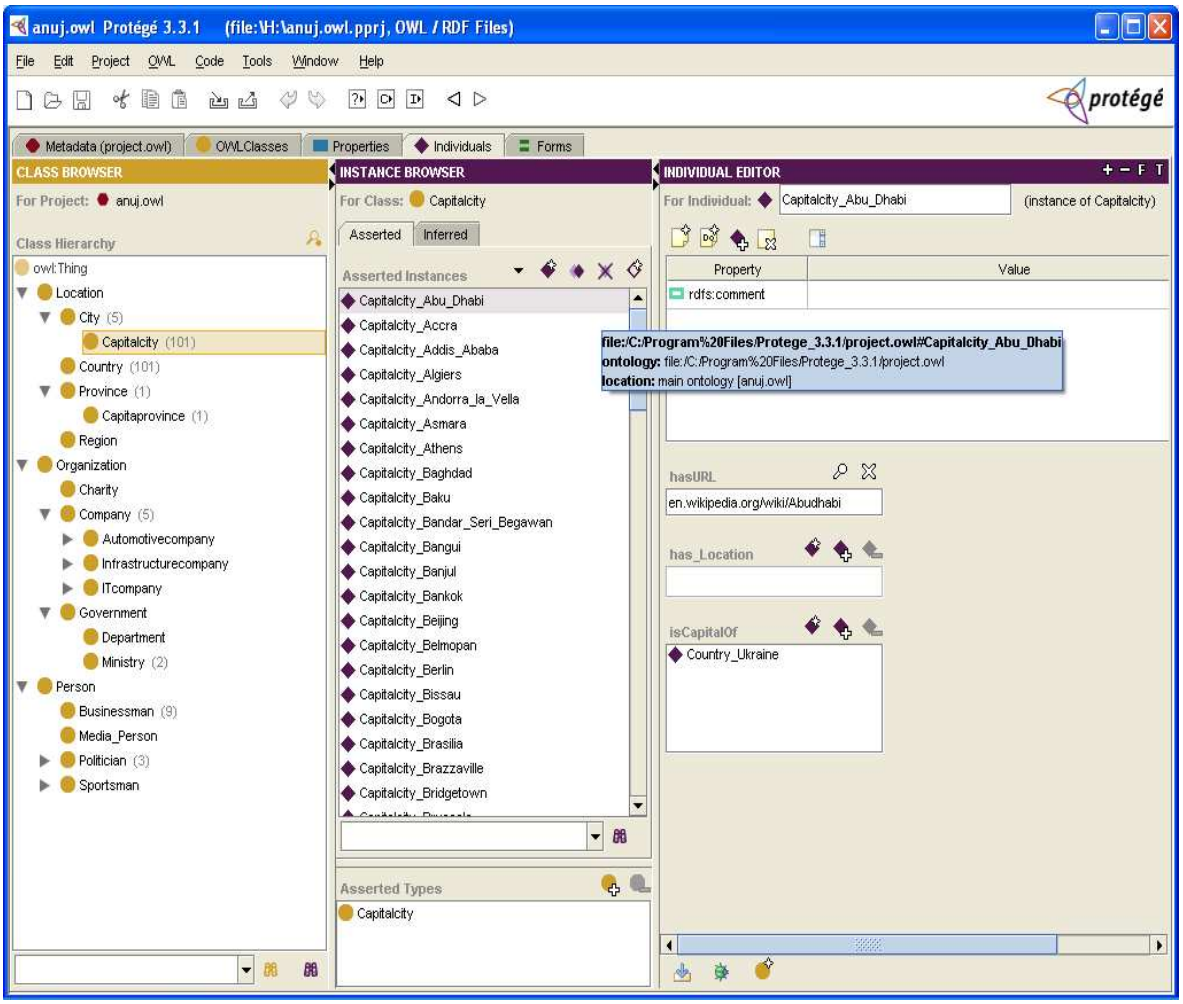


Figure 5.2: Protégé Interface showing Ontology

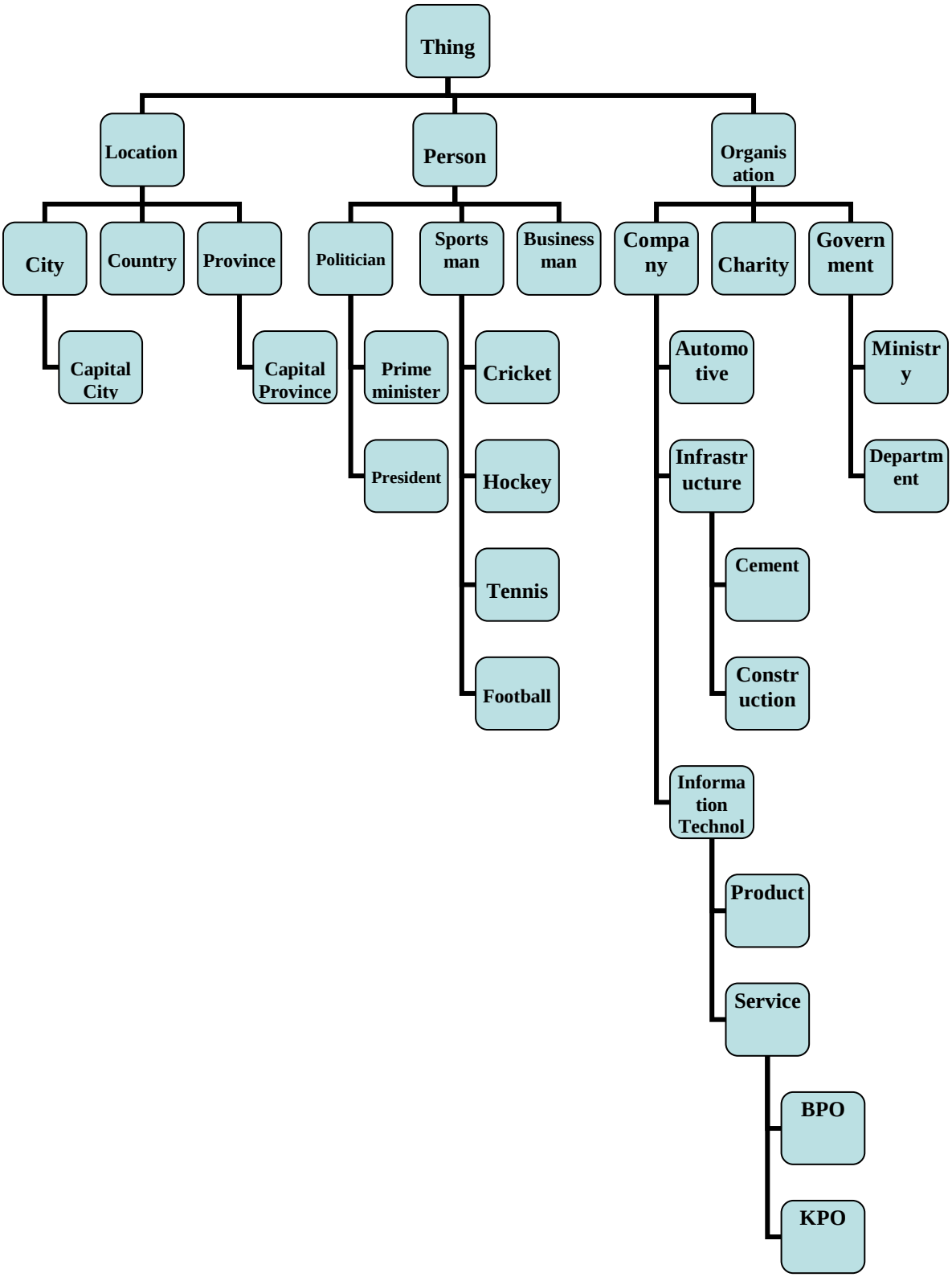


Figure 5.3: Ontology Class Hierarchy

5.2.2 Semantic Annotation

In this thesis we propose a *methodology* [21] to automatically annotate domain-specific information from large repositories (e.g. Web sites) with minimum user intervention. Semantic annotation is the process of inserting tags in the document, whose purpose is to assign semantics to the text between the opening and closing tags. The Semantic Web (SW) needs semantically annotated document to both enable better document retrieval and empower semantically-aware agents. Most of the current technologies are based on human centred annotation, very often completely manual. Manual annotation is difficult; time consuming and expensive .Convincing millions of users to annotate documents for the Semantic Web is difficult and requires a world-wide action of uncertain outcome.

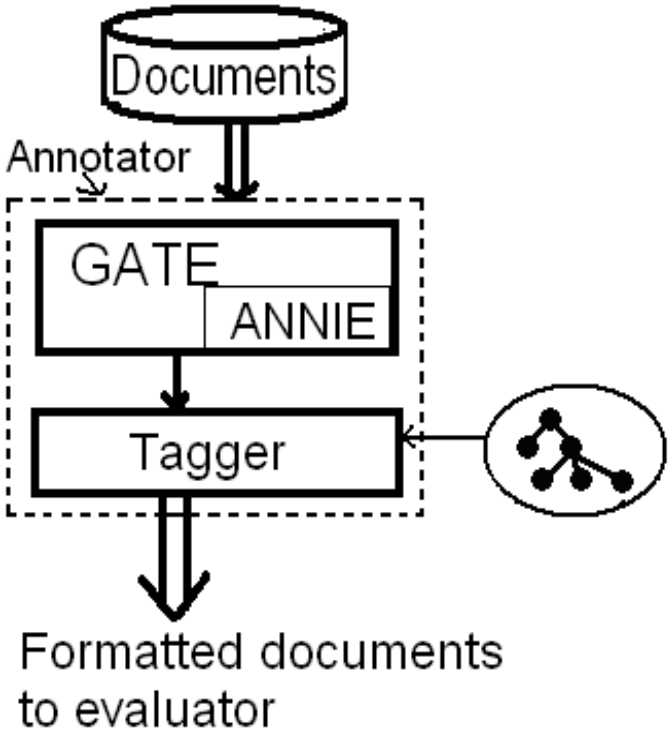


Figure 5.4: Information Extraction using ANNIE

The detailed view of the annotator is shown in the figure. It has a component GATE[9] (General Architecture for Text Engineering). GATE is an open source project. It includes modules for text engineering. It supports both language and dictionary based named entity extraction. It finds out

the common and proper nouns from a given text. Once the documents are submitted to GATE its ***ANNIE*** module (A Nearly New Information Extractor) gives a list of all the Named Entities (common and proper nouns) found by it. Tagger has a job of creating a mark-up in the document. This mark up stores the semantic information about the Named Entity in the document. Though there are many language standards defined for this type of mark-up but for our purpose and to keep things simple we use the 'class' attribute of the HTML tags to store the this semantic information. Be the need arise for the use of this attribute for some other purpose (*commonly css, cascading style sheets*) it can be replaced by a mark-up like rdfa. The tagger iterates the list of Named Entities given by ANNIE. If the current Named Entity is present in the Ontology it creates the mark up for this Named Entity in the document. If it is an instance of a class (say c1) in the ontology then a mark-up of the form is created. We prefix the class name by "m_" and suffix it by "_m" to prevent any collision for Cascading Style sheet (CSS) classes. It also adds some information from ontology to the title attribute of this tag. It has a sole purpose of enriching the information content of the document. Once the document has been annotated and the semantic information embedded, this document is passes on to other components for further processing. The formal method to annotate the document is as follows.

Method 1 Annotate

1. *We sort all the annotations given by Annie.*
    a. *For sorting we first check whether the annotations are overlapping or not*
    b. *If yes we leave them*
2. *we reverse the list of annotations sorted based on their start position in the document*
3. *for each annotation that is in the ontology*
    a. *we first add the end tag *
    b. *the we construct start <a class="m_class_m" title="Type:class">*

5.2.3 Ontology Population

The tool presented in this thesis also helps in semi automatically population of ***o****ntology*[20] .Since the portal involves the continuous flow of data ,so lots of new instances are identified everyday so to add these instances into the ontology we have developed an approach which involves minimum user intervention at administrator end. As the documents are processed by annotator some new entities are generated and administrator is presented with these entities. The tool shows the drop down box showing the ontology class hierarchy to which that instance can be added.

We find out all the annotations using ***G****ATE* [9] and present the user with them. GATE supports named entity extraction based on dictionary as well as grammar. By default GATE has grammars and dictionary defined for the three top level type that we use in our ontology. GATE can be extended to include more types also. The tentative type of the annotations is provided from ANNIE using its gazetteer lists. The document is first annotated temporarily for all the Named Entities presented by the ANNIE irrespective of whether they are present in the ontology or not. A button is created for each annotation. When user clicks the button for a specific annotation the list classes is shown. If the Named Entity is already present in the ontology, the name of its class is shown in different colour. User is also informed about the class of that annotation as given by ANNIE. Once the user selects the class for that Named Entity it is added to the ontology. The Gazetteer list of gate is also updated. Further User also has option of selecting any piece of text from the document and adds it as an instance in the ontology. This is done to enable user to add the Named Entity that ANNIE has failed to recognize.

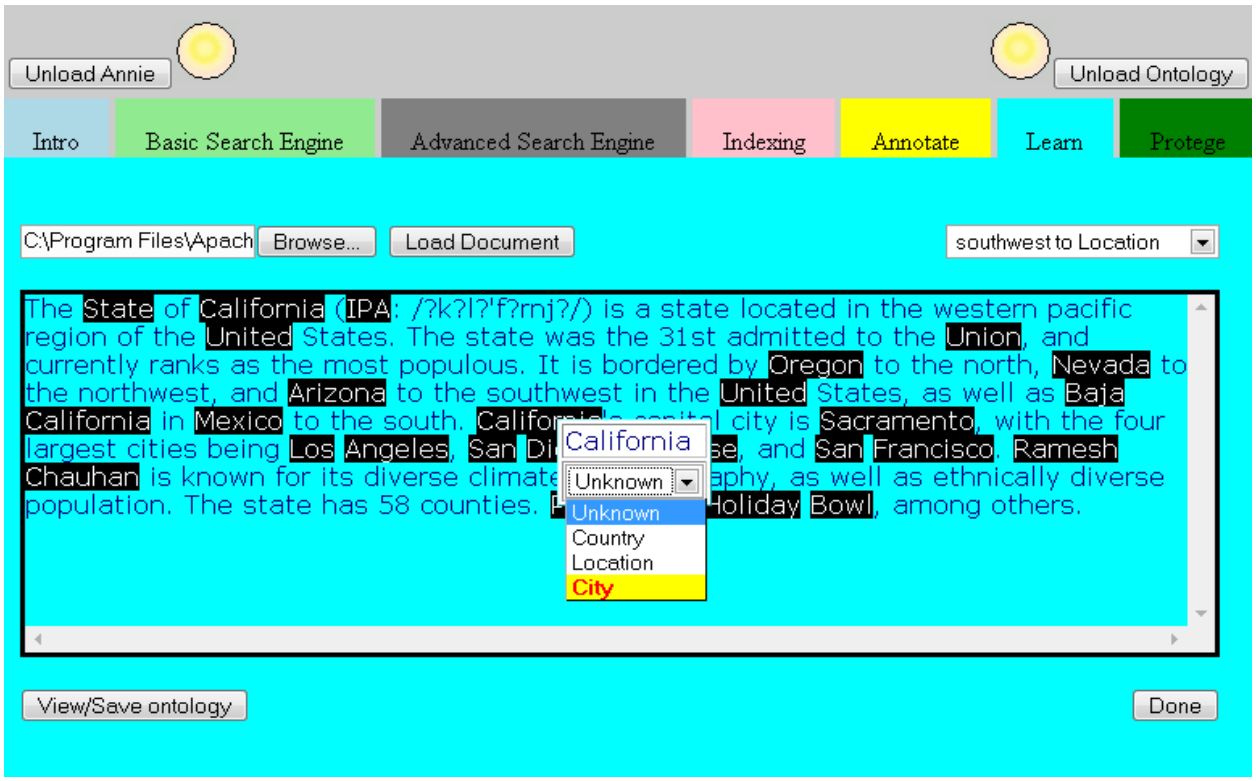


Figure 5.5: Ontology Population

6. IMPLEMENTATION DETAILS

There are two main modules in the system. First is ***OntologyManager***. It provides all the functions related to ontology. All other modules call functions of this module whenever they require any information related to ontology. The second module ***is AnnieManager/Named Entity Extractor***. It's an interface to GATE's ANNIE plug-in. It provides functions that are used to find annotations from the documents. The figure given shows the different modules of the tool.

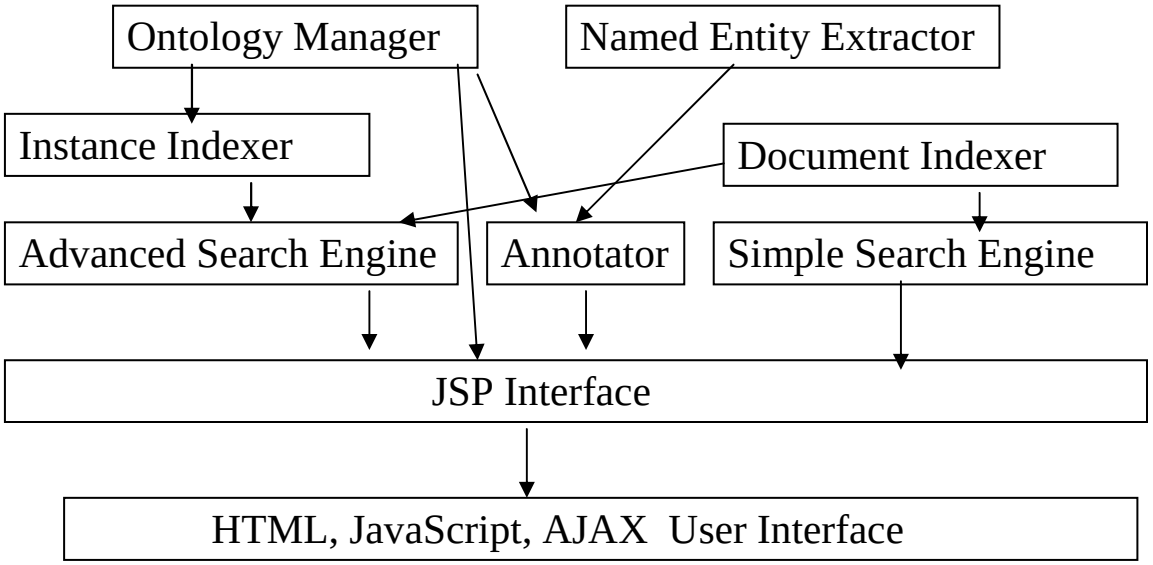


Figure 6.1: Functional Diagram of Tool

6.1 OntologyManager

It has been implemented as a single *OntologyManager.java* class. It provides all the functions related to ontology. All other modules call functions of this module whenever they require any information related to ontology. It uses protégé OWL API to read/save ontology from the file. It also initializes many data-structures which are used by other modules.

1. Subclass Hash table.

   Structure

Class Name	Reference to list of subclasses

2. Instance Hash Table.

   Structure

Instance Name	Reference to Instance Object

3. Class Hash Table

   Structure

Instance Name	Reference of the Class

6.2 Named Entity Extractor

It is wrapper around GATE's ANNIE plugin. It does the task of Initiliazing GATE, and loading ANNIE. It also provides functions to set the file to be annotated and to find annotations of that file. It provides a simple Iterator type interface to access Named entities of the set document. It has been Implemented as *AnnieManager.java*

6.3 Annotator

It has been implemented as another java file. *myAnnotate.java*. It calls the functions of Ontology Manager and the AnnieManager class. It provides functions to annotate document. It is called while population of ontology and annotating the document. It reads the files from uploaded Documents folder under tomcat root and extracts texts from them. Right now it can only read html and text files.

6.4 Instance Indexer

It gets all the instances in the ontology from Ontology Manager. Creates a Index of them using Lucene. It saves the index in OntoIndex folder under Tomcat folder.

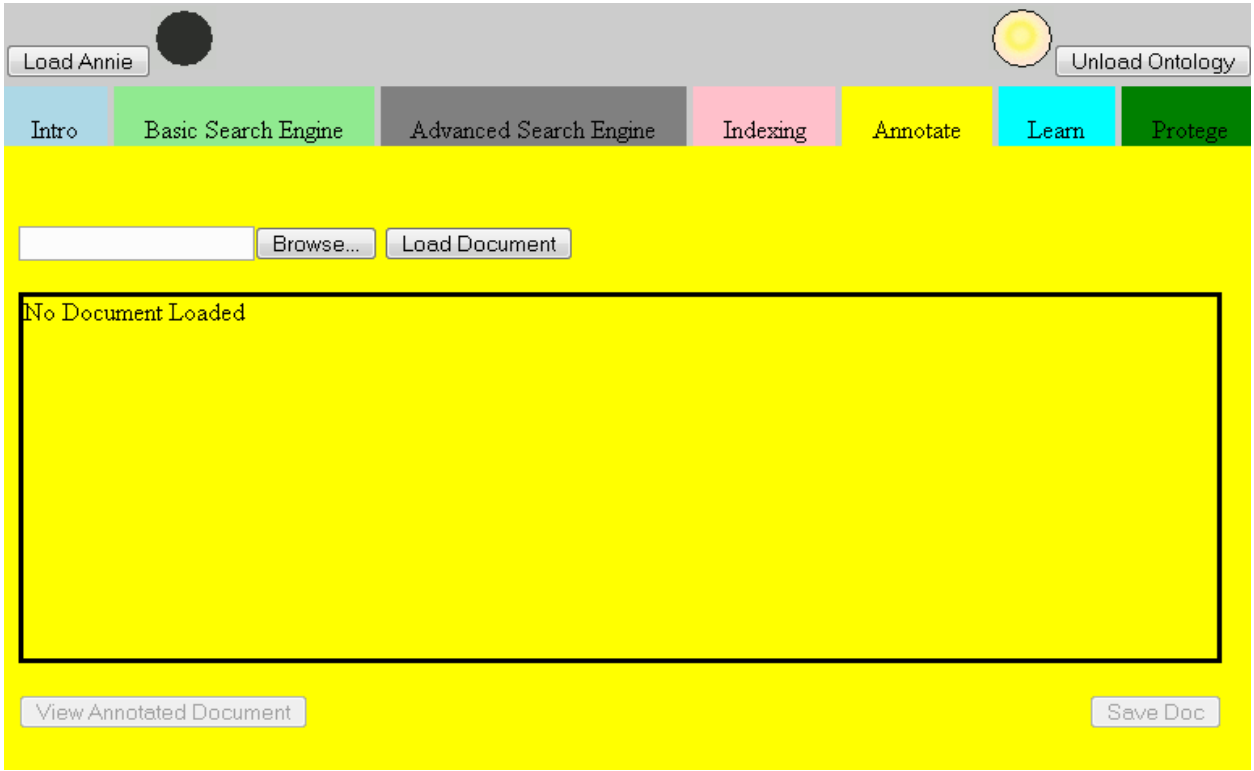
6.5 Document Indexer

It has been implemented as a multithreaded java class. It reads the documents from uploadedDocuments folder and adds each document one by one to the index. It also creates a queue of the names of document being indexed which is read by the GUI for displaying.

6.6 Graphical User Interface

Application such as this requires a very rich Graphic User Interface. GUI consists of tabs each from simple search, advance search, Indexing, Annotation and Ontology population.

Annotation Tab



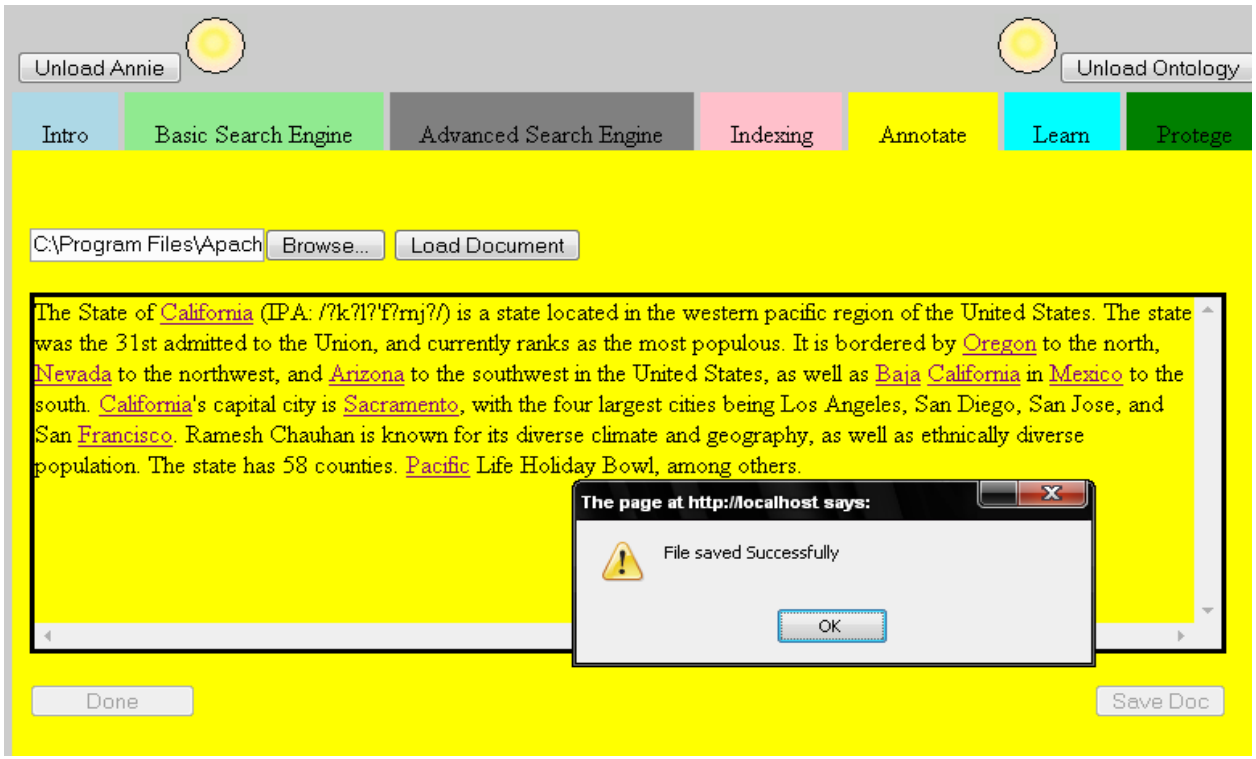
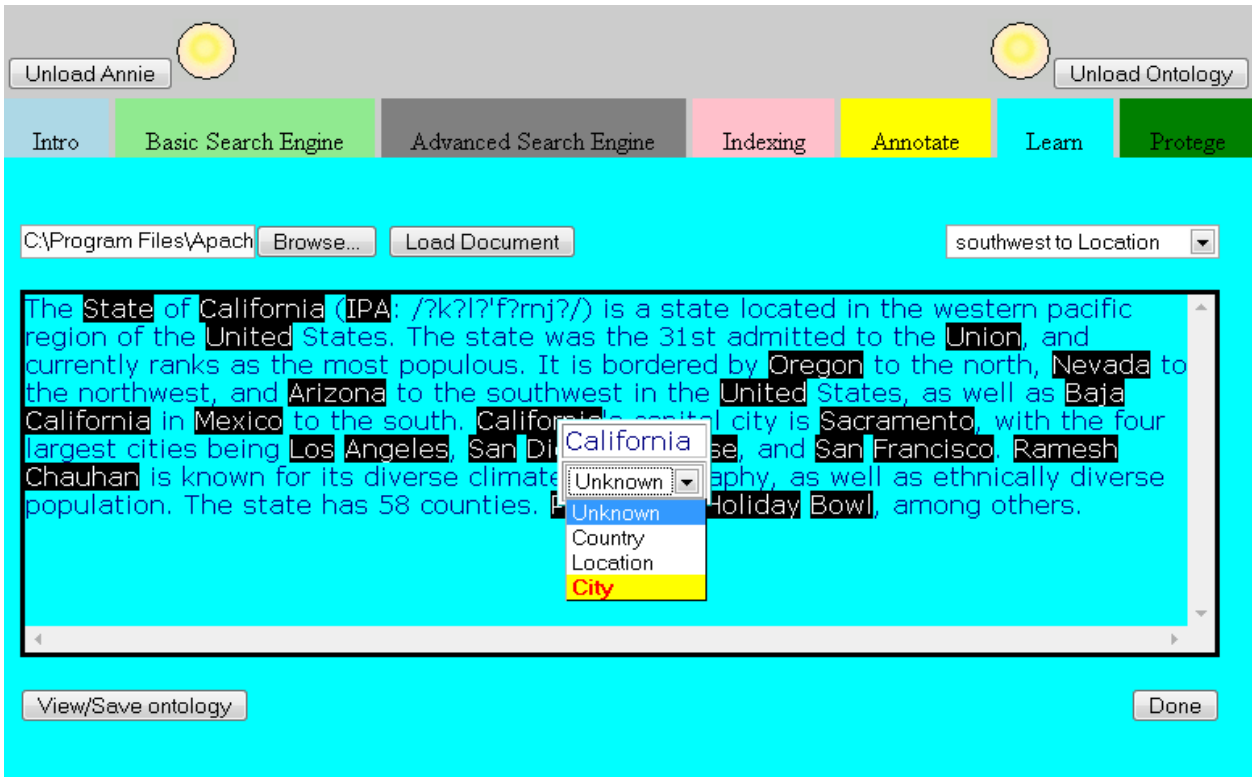


Figure 6.2: Annotation Interface of Tool

Ontology Population Tab



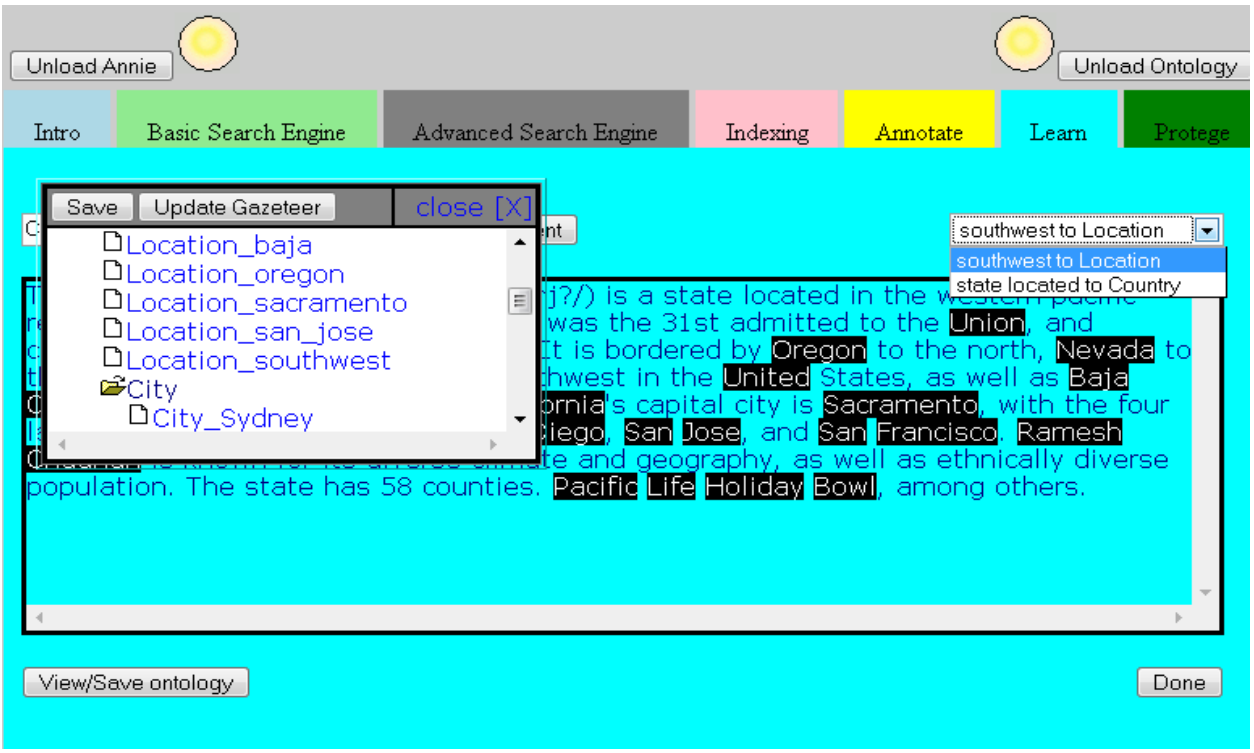


Figure 6.3: Ontology Population Interface of Tool

Web Protégé

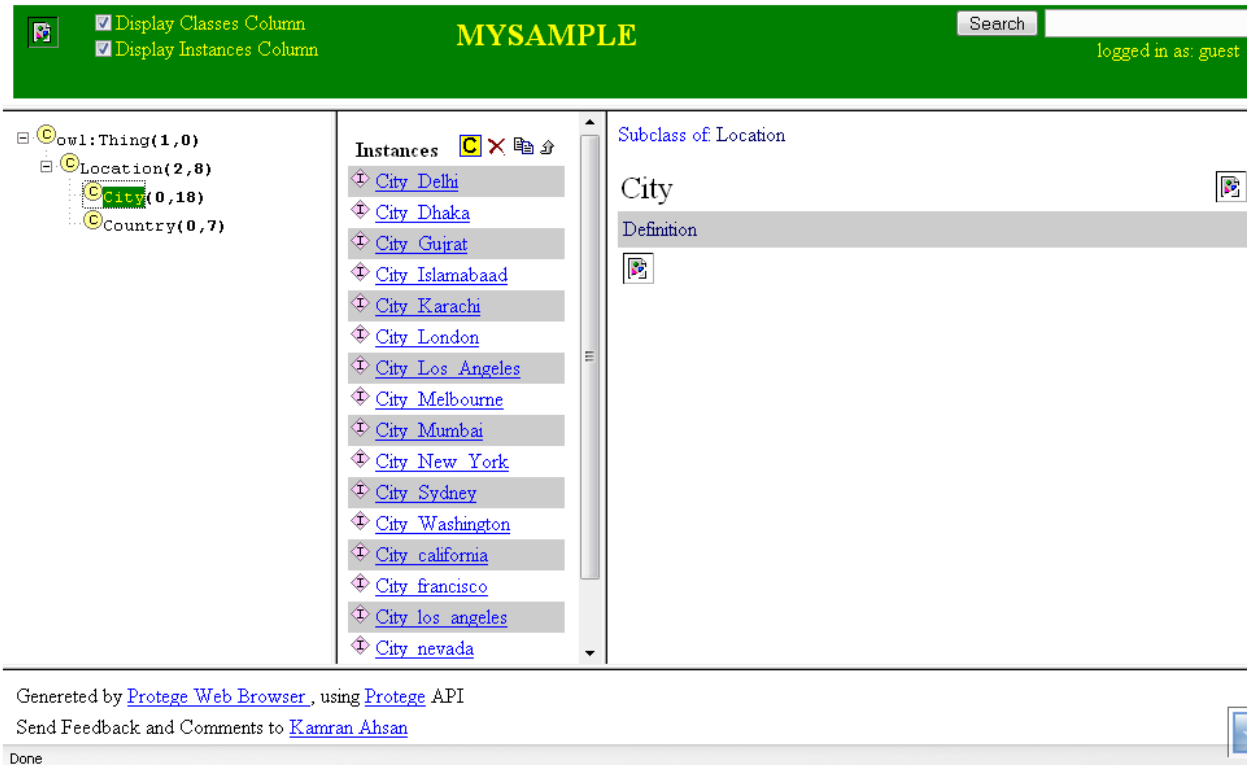


Figure 6.4: Web Interface of Protege

7. CONCLUSION & FUTURE WORK

All above algorithms and tool have been implemented in java as the National Panchayat Portal is also developed using java. We use a combination of java [22], jsp and AJAX to create this tool. It provides a very good graphic user interface with all sorts of pop-up menus, trees and custom controls etc. We also integrated web protégé [18] with the application for free form editing of ontology.

Our approach makes use ontology to represent concepts and we use available tools and techniques of semantic web to improve information retrieval processes. The tool we have developed provides integrated framework for semantic annotation , semi automatic ontology population, which can further be extended to complete semantic web infrastructure including semantic search.

This work was appreciated a lot by NIC people. This tool will also be used as a prototype for the plug-in for the National Panchayat Portal to make it as India's first ***S**emantic **P**ortal*[23]. We also received a certificate from them. We are really thankful for the help and support by NIC staff.

Many issues regarding scalability, securities etc. are yet to be addressed. Below is a list for possible future extensions for this project.

Future Work

1. Co-reference Resolution
2. Relational to RDF conversion
3. Use of Attributes in semantic annotation
4. Ontology Evolution
5. Using Different Ontologies on Same Domain.

8. PUBLICATION FROM THESIS

During the period of working over this project we interacted with International community working on semantic web. We discussed our approach for developing semantic web infrastructure with them and collected the reviews and worked over the suggestion send to us. Two research papers have been accepted in International conferences for presentation and will be published in their proceedings. Also we have communicated a journal paper, so that our work can be recognised and validated .These papers presents the methods and algorithm we have developed with detailed tool architecture that we have developed during the course of the project. The details of publications are as follows:

1. Conference Name **: "Semantic E-business and enterprise computing" SEEC 2008**
URL **: http://www.seec.eu/about.html**.
Paper Title **: "An Approach to Semantic Web"**
Authors **: Anuj Kumar, Nitin Nizhawan, Daya Gupta.**
Location **: Kerala, India.**
Conference Date **: Sept 17-19 2008.**

.

2. Conference Name **: "Semantic Web & Web Services" SWWS 2008.**
URL **: http://www.world-academy-of-science.org/worldcomp08/**
Paper Title **: "Semantic Search and Annotation Tool"**
Authors **: Anuj Kumar, Nitin Nizhawan, Daya Gupta.**
Location **: Las Vegas, USA.**
Conference Date **: July 14-17 2008.**

3 Journal Details : We have also worked on the comments and reviews received from different conferences and written a journal paper titled"***M****ethodology for* ***H****TML to* ***S****emantic* ***Web***" which has been communicated to the journal of Data Semantic. This journal paper describes the complete approach with some well defined articulated steps for converting the current web to semantic web.

REFERENCES

[1] Tim Berners-Lee. Realising the Full Potential of the Web. W3C Document, URL http://www.w3.org/1998/02/Potential.html, Dec 1997.

[2] Tim Berners-lee. Semantic Web Road Map. W3C Design Issues. URL http: //www.w3.org/DesignIssues/Semantic.html, Oct. 1998.

[3]. A. Banerji, C. Bartolini, D. Beringer, V. Chopella, K.Govindarajan, A. Karp, H. Kuno, M. Lemon, G. Pogossiants, S. Sharma, S. Williams: Web Services Conversation Language (WSCL), HP, 2001.

[4].Staab, S., Studer, R. editors: "Handbook on Ontologies", Springer, 2004.

[5]. D. Brickley and R.V. Guha, Resource Description Framework (RDF)Schema Specification 1.0, W3C Candidate Recommendation, March 27, 2000.

[6]. Martin Hepp, " Possible Ontologies: How Reality Constrains the Development of Relevant Ontologies", 2007, IEEE Computer society.

[7] "Protégé", http://protege.stanford.edu.

[8] Gobinda G. Chowdhury "Natural Language Processing",Dept. of Computer and Information Sciences University of Strathclyde, Glasgow G1 1XH, UK .

[9] "GATE, General Architecute for Text Engineering", http://www.gate.ac.uk

[10] Ramon F. Brena, Eduardo H. Ramirez, "A Soft Semantic Web", IEEE 2006.

[11] Rada F. Mihalcea, Silvana I. Mihalcea, "Word Semantics for Information Retrieval: Moving One Step Closer to the Semantic Web".

[12] "WordNet", http://wordnet.princeton.edu/

[13] Matthew Horridge,Holger Knublauch, Alan Rector, Robert Stevens, Chris Wroe, "A Practical Guide To Building OWL Ontologies Using The Prot´eg´e-OWL Plugin and CO-ODE Tools", The University Of Manchester.

[14] Atanas Kiryakov, Borislav Popov, Damyan Ognyanoff, Dimitar Manov, Angel Kirilov, Miroslav Goranov," Semantic Annotation, Indexing, and Retrieval", Ontotext Lab, Sirma AI EOOD, 138 Tsarigradsko Shose, Sofia 1784, Bulgaria.

[15] Trent Apted , Judy Kay," Automatic Construction of Learning Ontologies" , Proceedings of the International Conference on Computers in Education (ICCE'02) 2002 IEEE.

[16]. Xu Binfeng, Luo Xiaogang, Peng Chenglin and Huang Qian, "Based on Ontology: Construction and application of Medical Knowledge Base", 2007 IEEE/ICME Inrenational conference on complex medical engineering.

[17] Gupta D. Jat K., "Knowledge Representation Using Ontology and Relational database to RDF Converter", *EISWT 2008.*

[18] "Protégé Web Interface", http://protege.cim3.net/cgi-bin/wiki.pl?ProtegeWebBrowser.

[19] Brooke Abrahams. Wei Dai," Architecture for Automated Annotation and Ontology Based Querying of Semantic Web Resources", Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05).

[20] Kumar A. Nizhawan N. Gupta D."Semantic Search and Annotation Tool(SSAT)".SWWS'08.

[21] Kumar A. Nizhawan N. Gupta D."An approach to Semantic web", SEEC 2008.

[22] Jayson Falkner, Kevin Jones, "Servlets and Java ServerPages", Addison-Wesley, 2000

[23] Matt Perry, Eric Stiles, "SEMPL: A SEMantic PortaL" 2004

APENDIX A

AN INTRODUCTION TO GATE

(General Architecture for Text Engineering)

URL: http://gate.ac.uk/

GATE is an infrastructure for developing and deploying software components that process human language. GATE helps scientists and developers in three ways:

1. by specifying an architecture, or organizational structure, for language processing software;
2. By providing a framework or class library that implements the architecture and can be used to embed language processing capabilities in diverse applications;
3. By providing a development environment built on top of the framework made up of convenient graphical tools for developing components.

GATE can be thought of as a Software Architecture for Language Engineering

Language Engineering (LE) may be defined as:

. . . the discipline or act of engineering software systems that perform tasks involving processing human language. Both the construction process and its outputs are measurable and predictable. The literature of the field relates to both application of relevant scientific results and a body of practice.

GATE as an architecture suggests that the elements of software systems that process natural language can usefully be broken down into various types of component, known as resources. Components are reusable software chunks with well-defined interfaces, and are a popular architectural form, used in Sun's Java Beans and Microsoft's .Net, for example. GATE components are specialised types of Java Bean, and come in three flavours:

• LanguageResources (LRs) represent entities such as lexicons, corpora or ontologies.

• ProcessingResources (PRs) represent entities that are primarily algorit- -hmic, such as parsers, generators or ngram modelers.

• VisualResources (VRs) represent visualisation and editing components that participate in GUIs.

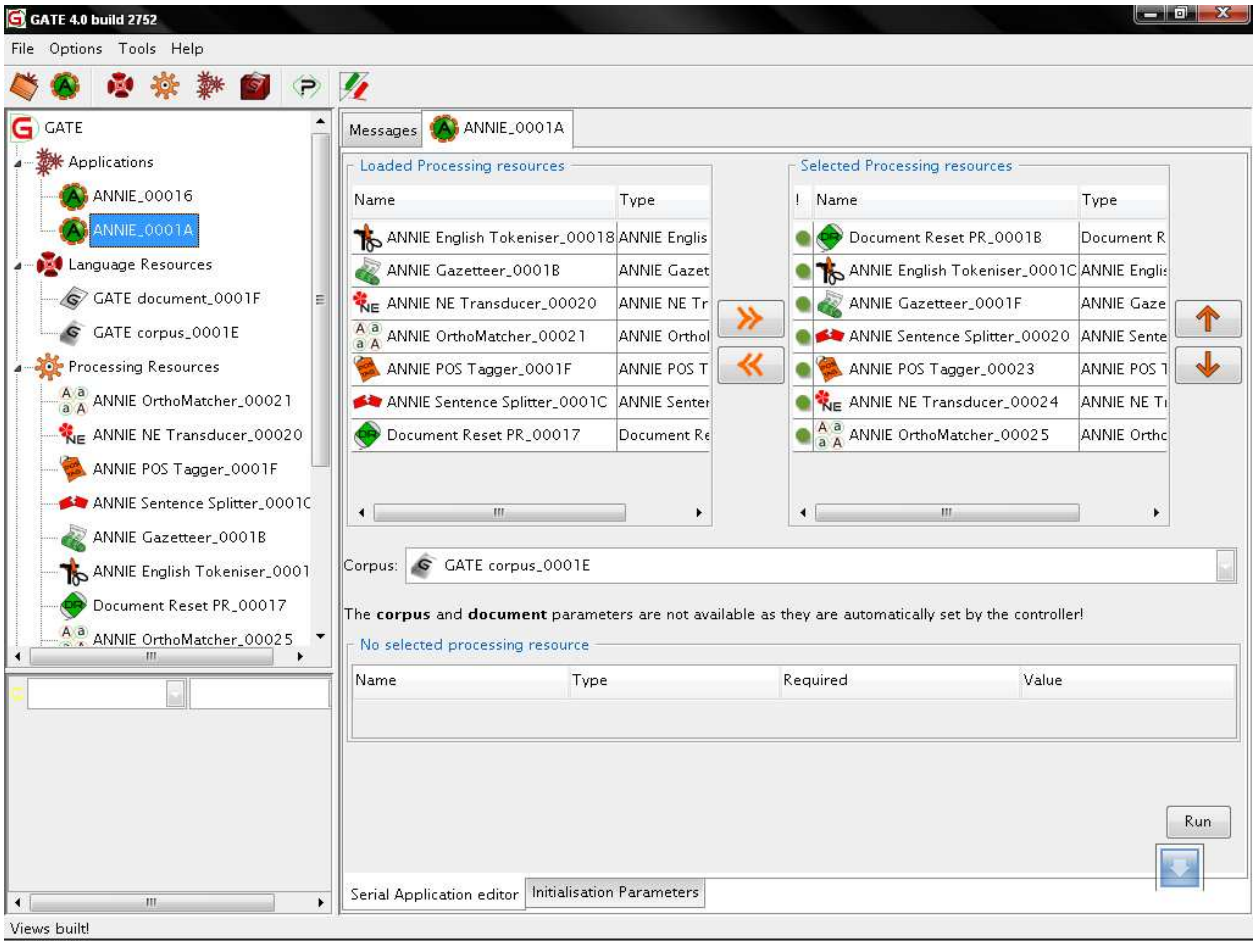


Figure. GATE with ANNIE loaded

Built-in Components:

ANNIC

ANNotations In Context: a full-featured annotation indexing and retrieval system designed to support corpus querying and JAPE rule authoring. It is provided as part of an extention of the Serial Datastores, called Searchable Serial Datastore.

Machine learning API

A brand new machine learning layer specifically targetted at NLP tasks including text classification, chunk learning (e.g. for named entity recognition) and relation learning.

Ontology API

A new ontology API, based on OWL In Memory (OWLIM), which o_ers a better API, revised ontology event model and an improved ontology editor to name but few.

OCAT

Ontology-based Corpus Annotation Tool to help annotators to manually annotate documents using ontologies.

Alignment Tools

A new set of components (e.g. CompoundDocument, AlignmentEditor etc.) that help in building alignment tools and in carrying out cross-document processing.

New HTML Parser

A new HTML document format parser, based on Andy Clark's NekoHTML. This parser is much better than the old one at handling modern HTML and XHTML constructs, JavaScript blocks, etc., though the old parser is still available for existing applications that depend on its behaviour.

Ontotext Japec Compiler

Japec is a compiler for JAPE grammars developed by Ontotext Lab. It has some limitations compared to the standard JAPE transducer implementation, but can run JAPE grammars up to five times as fast.

Ontogazetteer

The Ontogazetteer, or Hierarchical Gazetteer, is an interface which makes ontologies "visible" in GATE, supporting basic methods for hierarchy management and traversal. In GATE, an ontology is represented at the same level as a document, and has nodes called classes.

Ontogazetteer Editor

This is for editing the class hierarchy of an ontology. it provides storing to and loading from RDF/RDFS, and provides load/edit/store of the class hierarchy of an ontology.

JAPE

Java Annotation Patterns Engine. JAPE provides finite state transduction over annotations based on regular expressions. JAPE is a version of CPSL
– Common Pattern Specification Language.

ANNIE: a Nearly-New Information Extraction System

GATE was originally developed in the context of Information Extraction (IE) R&D, and IE systems in many languages and shapes and sizes have been created using GATE with the IE components that have been distributed with it.
ANNIE components form a pipeline which appears in figure.

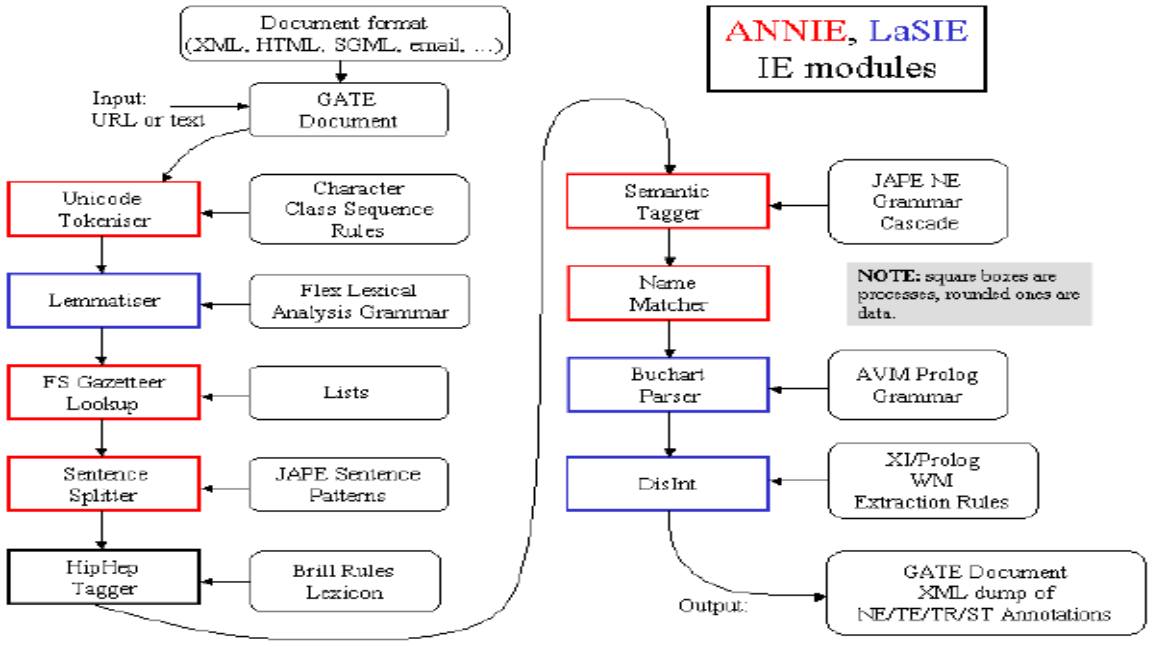


Figure. Annie data flow diagram

Tokeniser

The tokeniser splits the text into very simple tokens such as numbers, punctuation and words of different types. For example, we distinguish between words in uppercase and lowercase, and between certain types of punctuation.

Gazetteer

The gazetteer lists used are plain text files, with one entry per line. Each list represents a set of names, such as names of cities, organisations, days of the week, etc. An index file (lists.def) is used to access these lists.

GATE includes resources for common LE data structures and algorithms, including documents, corpora and various annotation types, a set of language analysis components for Information Extraction and a range of data visualisation and editing components. GATE supports documents in a variety of formats including XML, RTF, email, HTML, SGML and plain text. In all cases the format is analysed and converted into a single unified model of annotation.

APENDIX B

An Introduction to Protégé

URL: http://protege.stanford.edu.//

Protégé is the latest tool in an established line of tools developed at Stanford University for knowledge acquisition. Protégé has thousands of users all over the world who use the system for projects ranging from modelling cancer-protocol guidelines to modelling nuclear-power stations. Protégé is freely available for download under the Mozilla open-source license.

Protégé provides a graphical and interactive ontology-design and knowledge-base–development environment. It helps knowledge engineers and domain experts to perform knowledge-management tasks. Ontology developers can access relevant information quickly whenever they need it, and can use direct manipulation to navigate and manage an ontology. Tree controls allow quick and simple navigation through a class hierarchy. Protégé uses forms as the interface for filling in slot values. The **knowledge model** of Protégé-2000 includes support for classes and the class hierarchy with multiple inheritance; template and own slots; specification of pre-defined and arbitrary facets for slots, which include allowed values, cardinality restrictions, default values, and inverse slots, metaclasses and metaclass hierarchy. In addition to highly usable interface, two other important features distinguish Protégé from most ontology-editing environments: its **scalability** and **extensibility**.

One of the major advantages of the Protégé architecture is that the system is constructed in an open, modular fashion. Its component-based architecture enables system builders to add new functionality by creating appropriate plugins. The Protégé Plugin Library3 contains contributions from developers all over the world.

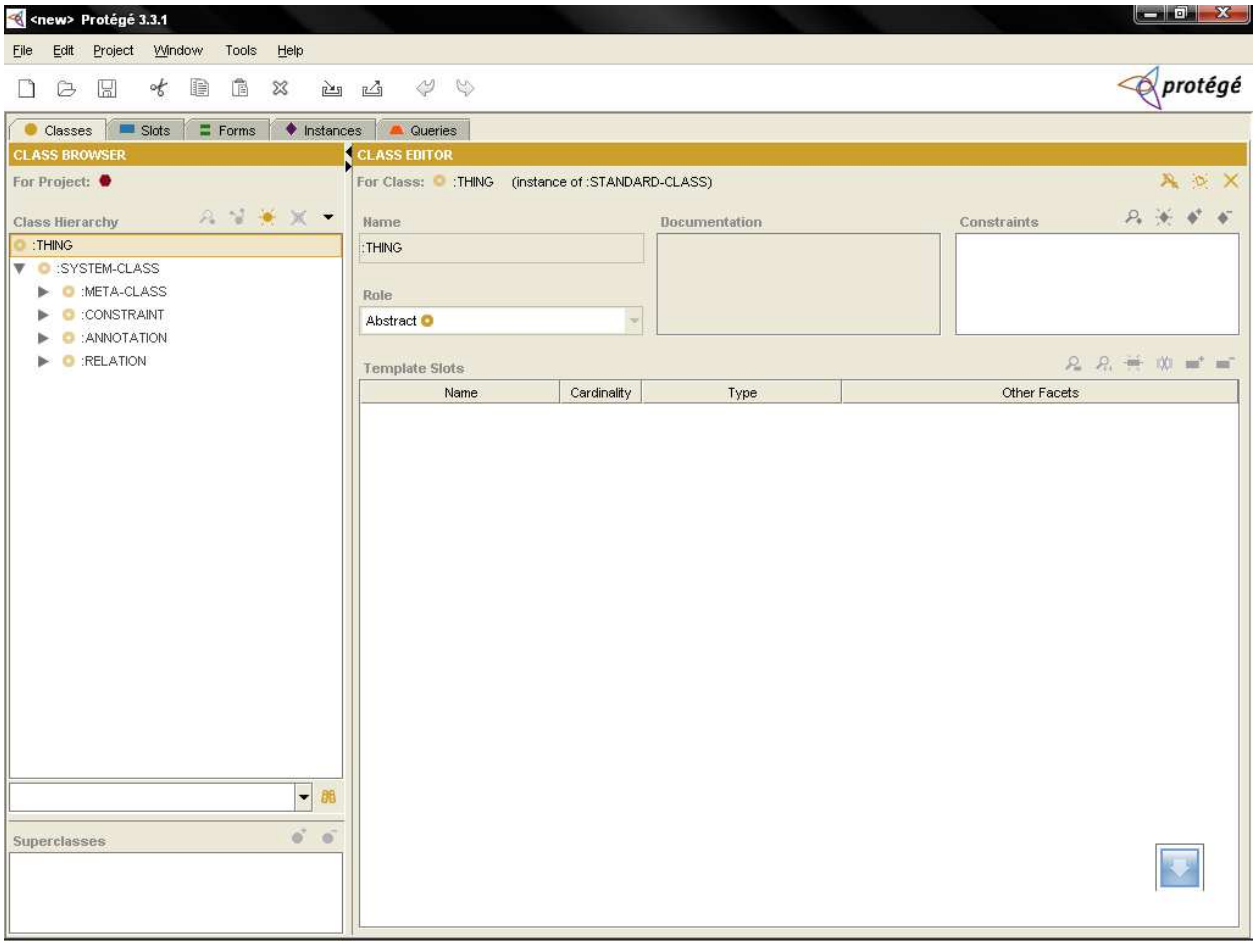


Figure. Protégé

The Protégé-OWL API is an open-source Java library for the Web Ontology Language (OWL) and RDF(S). The API provides classes and methods to load and save OWL files, to query and manipulate OWL data models, and to perform reasoning based on Description Logic engines. Furthermore, the API is optimized for the implementation of graphical user interfaces. The Protégé-OWL API is centred around a collection of Java interfaces from the model package. These interfaces provide access to the OWL model and its elements like classes, properties, and individuals.

The API is designed to be used in two contexts:

- For the development of components that are executed inside of the Protégé-OWL editor's user interface
- For the development of stand-alone applications (e.g., Swing applications, Servlets, or Eclipse plug-ins)

Protégé is a flexible, configurable platform for the development of arbitrary model-driven applications and components. Protégé has an open architecture that allows programmers to integrate plug-ins, which can appear as separate tabs, specific user interface components (widgets), or perform any other task on the current model. The Protégé-OWL editor provides many editing and browsing facilities for OWL models, and therefore can serve as an attractive starting point for rapid application development. Developers can initially wrap their components into a Protégé tab widget and later extract them to distribute them as part of a stand-alone application.

APPENDIX C

AN INTRODUCTION TO LUCENE

Apache Lucene is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform. Lucene is a free/open source information retrieval library, originally implemented in Java by Doug Cutting. It is supported by the Apache Software Foundation and is released under the Apache Software License. Lucene has been ported to programming languages including Delphi, Perl, C#, C++,Python, Ruby and PHP.

While suitable for any application which requires full text indexing and searching capability, Lucene has been widely recognized for its utility in the implementation of Internet search engines and local, single-site searching. Lucene itself is just an indexing and search library and does not contain crawling and HTML parsing functionality. The Apache project Nutch is based on Lucene and provides this functionality; the Apache project Solr is a fully-featured search server based on Lucene.

At the core of Lucene's logical architecture is a notion of a document containing fields of text. This flexibility allows Lucene's API to be agnostic of file format. Text from PDFs, HTML, Microsoft Word documents, as well as many others can all be indexed so long as their textual information can be extracted.